

# Introducción a Simulink con Arduino y Raspberry Pi

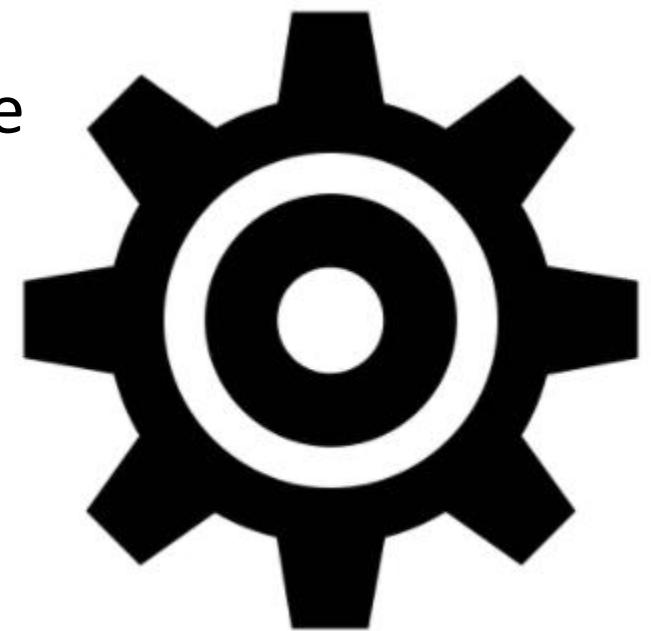
Juan Moreno García-Loygorri

Madrid, 20-21/MAY/2019

# Presentación del curso

# 1. Objetivos

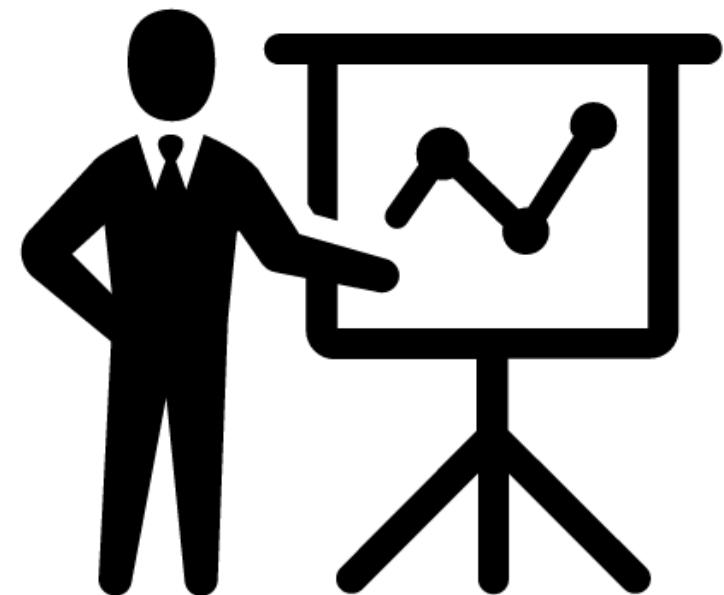
- Introducción a Simulink con Arduino y Raspberry Pi
  - Parte introductoria: conocimientos básicos de Simulink (Curso '101').
  - Parte más avanzada: uso de MATLAB con Simulink, sistemas dinámicos,
  - Control de placas Arduino y Raspberry PI
    - Ejemplos prácticos



## 2. Profesor

Juan Moreno García-Loygorri

- Dr. Ingeniero Teleco.
- Ferroviario (Metro de Madrid).
- Prof. asociado (ETSIST-UPM).
- [juanmorenogl@diac.upm.es](mailto:juanmorenogl@diac.upm.es)



# 3. Contenido del curso y plan (orientativo)

- 0. Introducción a MATLAB
- 1. Introducción a Simulink
  - 1. Entorno gráfico.
  - 2. Simulaciones.
  - 3. Bloques y señales.
  - 4. Visualización.
  - 5. Operadores matemáticos.
  - 6. Algoritmos básicos.
  - 7. Ayuda de MATLAB &Simulink.
- 2. Opciones avanzadas
  - 1. Uso de variables y datos

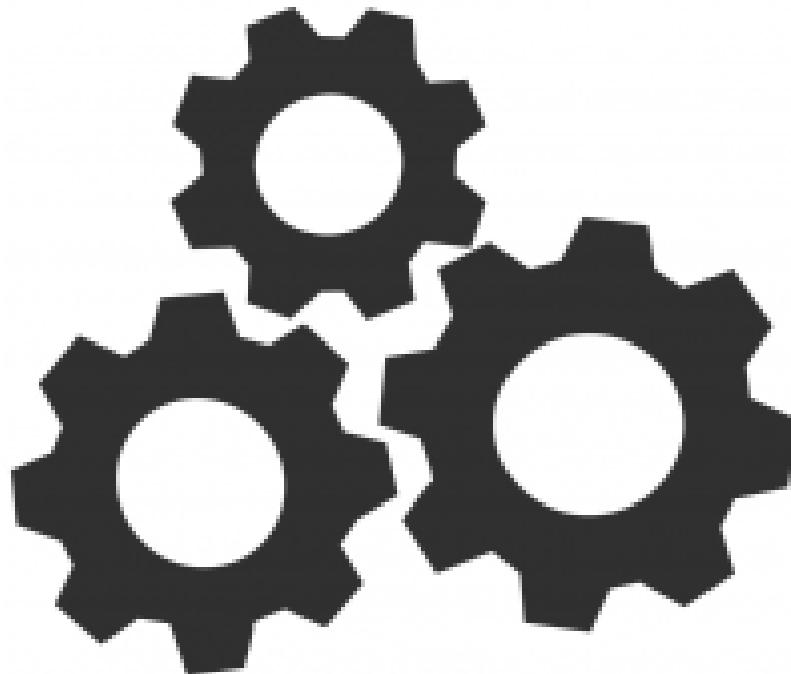
## Lunes

- de MATLAB en Simulink (y viceversa).
- 2. Código MATLAB en Simulink.
- 3. Sistemas dinámicos discretos y continuos. Concepto.
- 4. Configuración de MATLAB &Simulink para trabajar con Arduino y Raspberry Pi.

## Martes

- 1. Programación de Arduinos
  - 1. Placas Arduino. Descripción. Funcionamiento
  - 2. Proyecto de Arduino usando Simulink.
- 2. Programación de Raspberry Pi
  - 1. Introducción a la Raspberry Pi. Descripción. Funcionamiento.
  - 2. Proyecto con Raspberry Pi usando Simulink.

## 5. Metodología



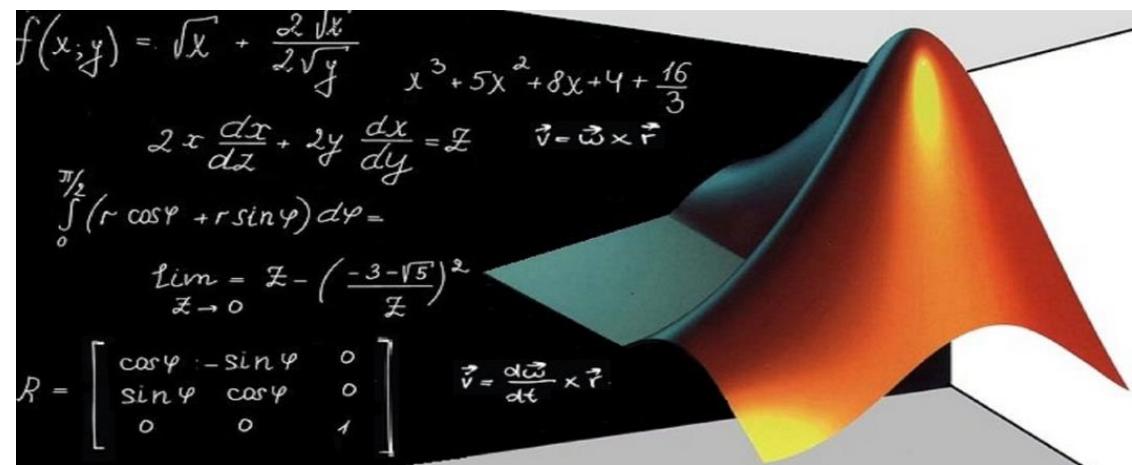
- Enfoque práctico.
  - Breves explicaciones teóricas.
  - Trabajo sobre el código.
  - Ejercicios.
- 1h50 – descanso – 1h50horas.

# ¡Empezamos!

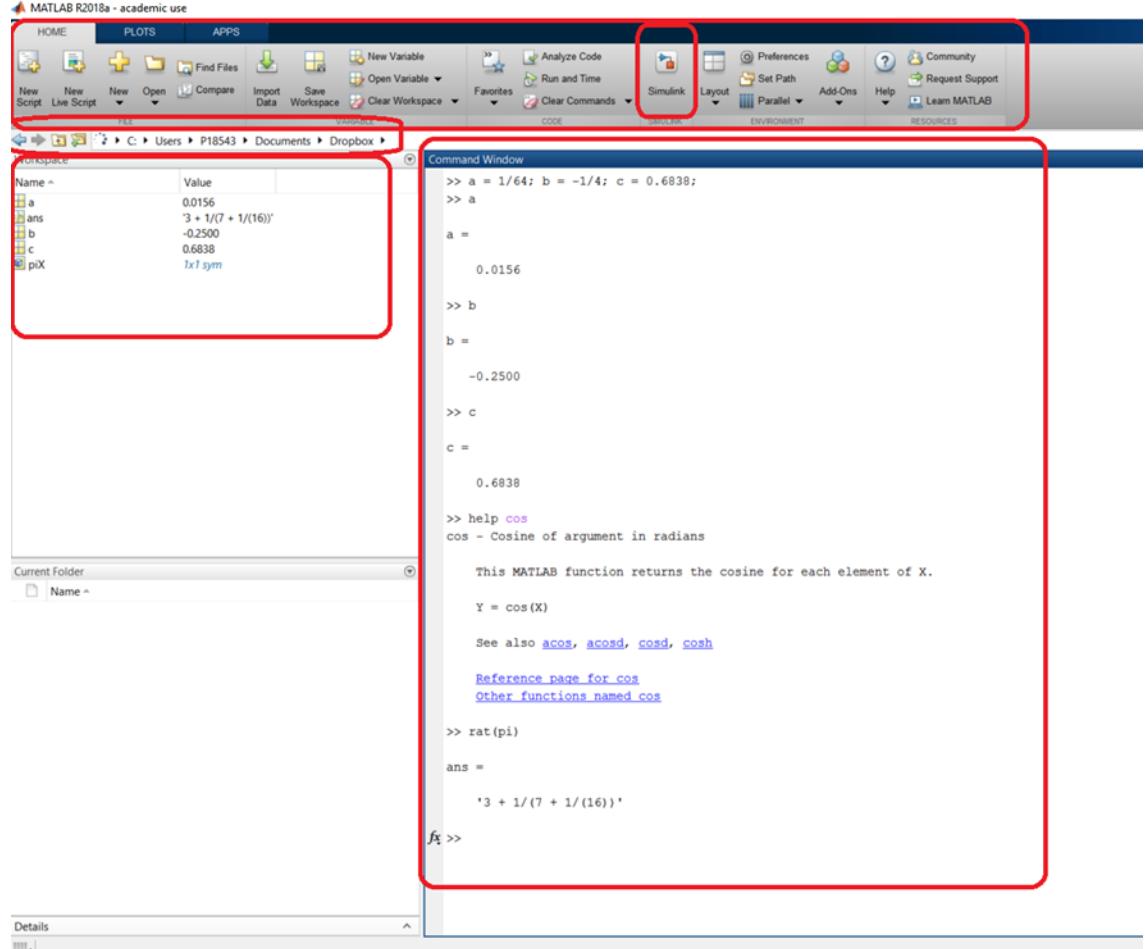
- ¿Dudas?
- Presentaciones de todos los participantes
- ¡Arrancamos MATLAB! ☺



# Módulo 0. Entorno MATLAB.



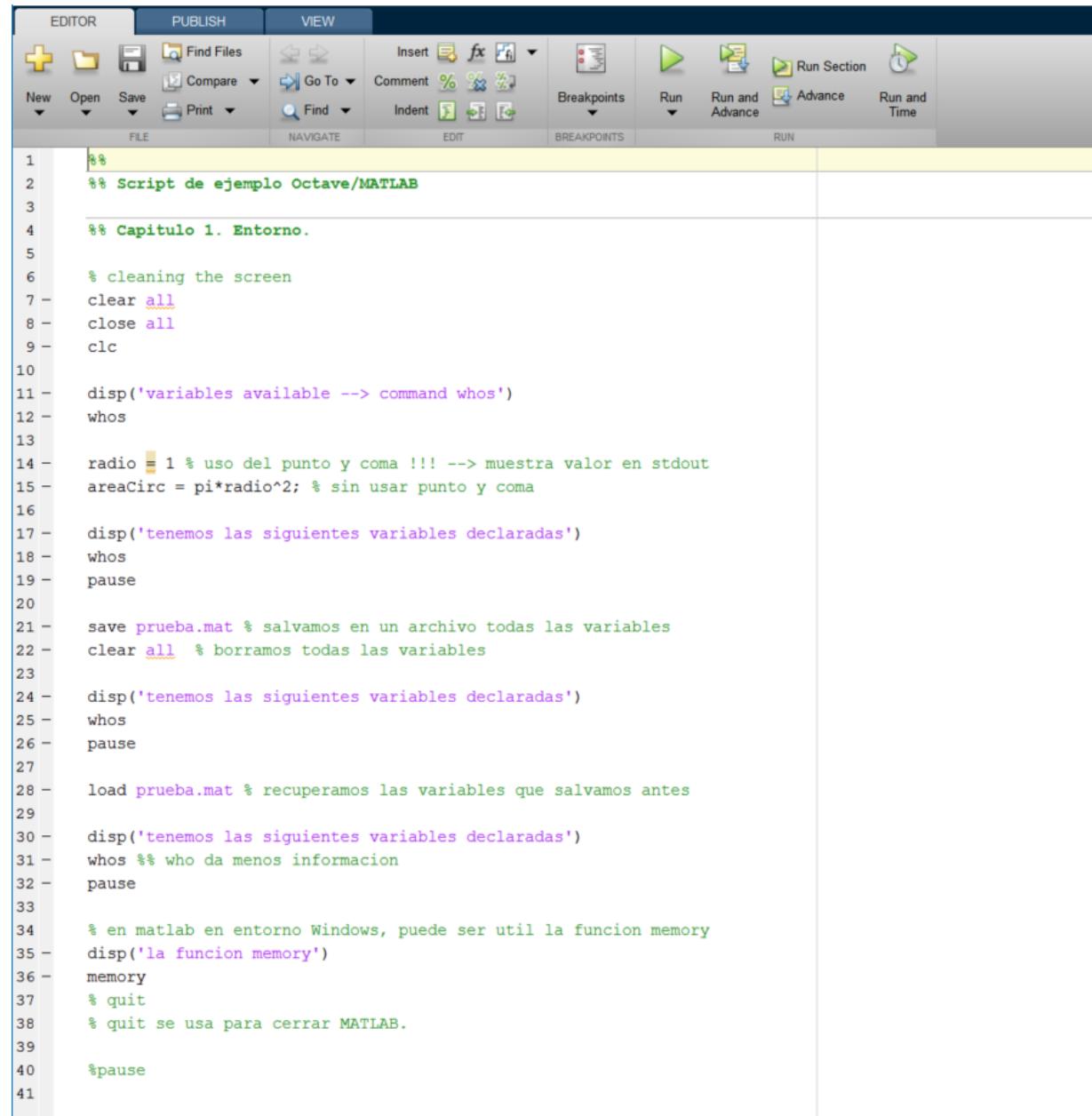
# 1. Entorno



- Línea de comandos.
- Directorio actual.
- Variables (workspace).
- Archivos.
- Menú:
  - *plots*
  - *home*
  - *apps*.

## 2. Editor

- Herramienta potente
- Se puede usar cualquier editor de texto
- Entorno integrado
- Otras opciones:
  - Debug.
  - Comparación
  - Identar.
  - Comentar.
  - (Live editor)



The screenshot shows the MATLAB Live Editor interface. The top menu bar includes 'EDITOR', 'PUBLISH', and 'VIEW'. Below the menu are various toolbar icons for file operations (New, Open, Save, Find Files, Compare, Print), navigation (Go To, Find, Indent), editing (Breakpoints, Run, Run and Advance, Advance, Run and Time), and breakpoints (Breakpoints, Run, Run and Advance, Advance, Run and Time). The main workspace displays a script with line numbers and MATLAB code:

```
1 %% Script de ejemplo Octave/MATLAB
2
3 %% Capítulo 1. Entorno.
4
5 % cleaning the screen
6 clear all
7 close all
8 clc
9
10 disp('variables available --> command whos')
11 whos
12
13 radio = 1 % uso del punto y coma !!! --> muestra valor en stdout
14 areaCirc = pi*radio^2; % sin usar punto y coma
15
16 disp('tenemos las siguientes variables declaradas')
17 whos
18 pause
19
20 save prueba.mat % salvamos en un archivo todas las variables
21 clear all % borramos todas las variables
22
23 disp('tenemos las siguientes variables declaradas')
24 whos
25 pause
26
27 load prueba.mat % recuperamos las variables que salvamos antes
28
29 disp('tenemos las siguientes variables declaradas')
30 whos %% who da menos informacion
31 pause
32
33 % en matlab en entorno Windows, puede ser util la funcion memory
34 disp('la funcion memory')
35 memory
36
37 % quit
38 % quit se usa para cerrar MATLAB.
39
40 %pause
41
```

### 3. Comandos básicos: entorno

exit	% para salir
ver	% versiones y toolboxes instaladas
;	% no muestra resultado
%	% comentarios a partir de aquí
...	% para partir en varias líneas

whos

who

whos –file myFile.mat

diary fileName	diary on	diary off
----------------	----------	-----------

# 4. Ayuda

stack overflow Questions Developer Jobs Tags Users [matlab] Log In Sign Up

Tagged Questions info newest featured frequent votes active unanswered

MATLAB is a high-level language and interactive programming environment for numerical computation and visualization developed by MathWorks. Questions should be tagged with either [tag:matlab] or [tag:octave], but not both, unless the question explicitly involves both packages. When using this tag, ...

Learn more... Top users Synonyms (5) matlab jobs

-1 votes 0 answers 7 views How to draw the neural network covariance function、Gibbs covariance function and Matern covariance function in Matlab? [on hold] How to draw the neural network covariance function、Gibbs covariance function and Matern covariance function in Matlab? While reading the thesis "Gaussian Processes for Machine Learning" written by C.E.... asked 3 hours ago yang 1 ● 1

matrix × 6217 image-processing × 5411 plot × 4190 arrays × 3638 matlab-figure × 2706 image × 2353

Ask Question

# 4. Ayuda

The screenshot shows the MATLAB Answers homepage. At the top, there's a navigation bar with links for Products, Soluciones, Educación, Soporte, Comunidad, and Eventos. Below that is a search bar with the placeholder "Buscar Answers". A dropdown menu labeled "Answers" is open, showing options like "Software de prueba". The main banner features the text "Get answers to your MATLAB and Simulink questions" and displays three statistics: 195.673 Questions answered, 104.198 Answers accepted, and 240.459 Members contributing.

Ask a Question

Go

Reddit

Answer a Question

The screenshot shows a Reddit page with a blue header bar. The title bar includes "Share question" and "MATLAB". Below the header, a sidebar lists filter options: activo, nuevo, subiendo, polémico, popular, con gold, and wiki. The main content area displays several posts from the r/matlab subreddit. The first post is a "ModPost" titled "Get an invitation code to the new r/matlab discord channel." with 10 upvotes. Other visible posts include "Submitting Homework questions? Read this.", "Quick Question Regarding a Model Rocket Altitude Estimation Program", "How to read in a .dat file into matlab?", and "Quick help with plotting streamlines.". Each post includes a small thumbnail, the number of upvotes, the user who posted it, and a link to the full post.

Web oficial Mathworks



# 4. Ayuda

> help nombreFuncion

```
>> help linspace
linspace - Generate linearly spaced vector
```

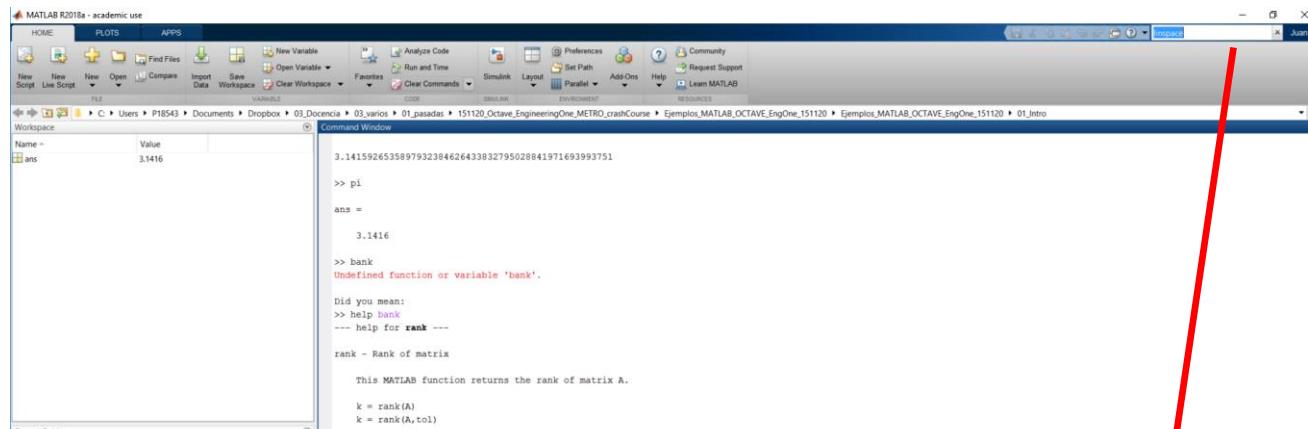
This MATLAB function returns a row vector of 100 evenly spaced points between  $x_1$  and  $x_2$ .

```
y = linspace(x1,x2)
y = linspace(x1,x2,n)
```

See also [colon](#), [logspace](#)

[Reference page for linspace](#)  
[Other functions named linspace](#)

> lookfor palabraClave



## 5.1. Buenas prácticas: principio KISS



- Filosofía básica de diseño.
- Simplicidad.
- Origen en industria aeronáutica (Skunk Works).
- No “inventar la rueda”.

## 5.2. Buenas prácticas: código

- Nombres de variables descriptivos.
- Funciones para cosas repetidas.
- No usar números mágicos.
- Comentarios sobre todo en partes oscuras.
- Utilizar funciones predefinidas de MATLAB.
- Vectorización → evitar bucles
- Buenas estructuras de datos.
- Precargar matrices.

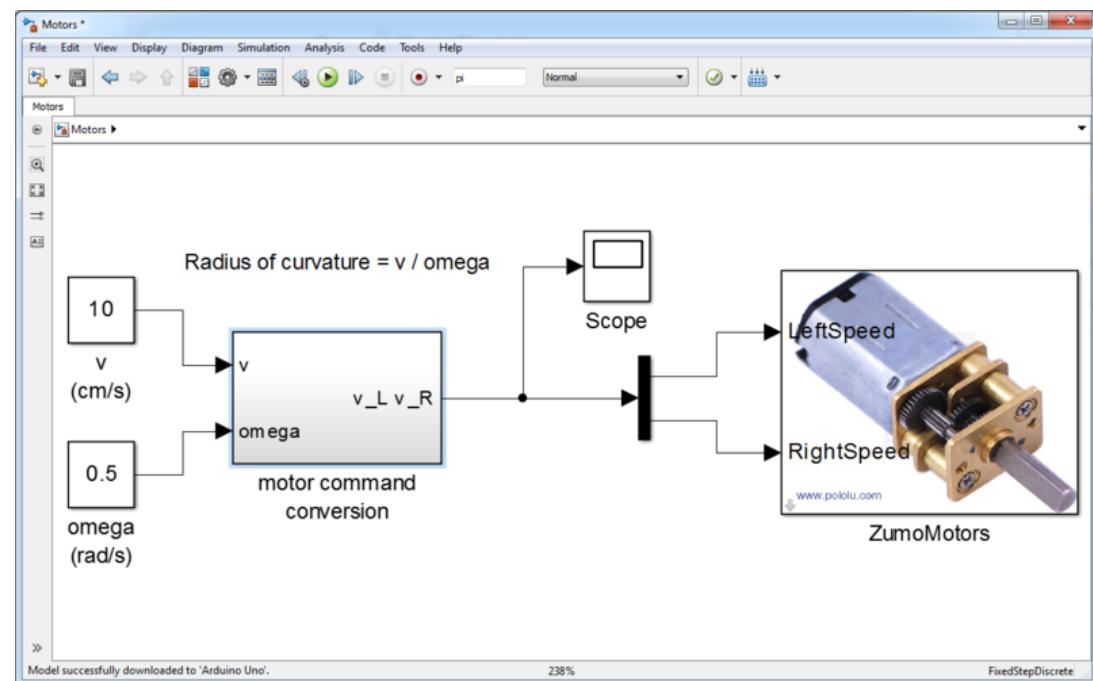


# Módulo 1. Introducción a Simulink.

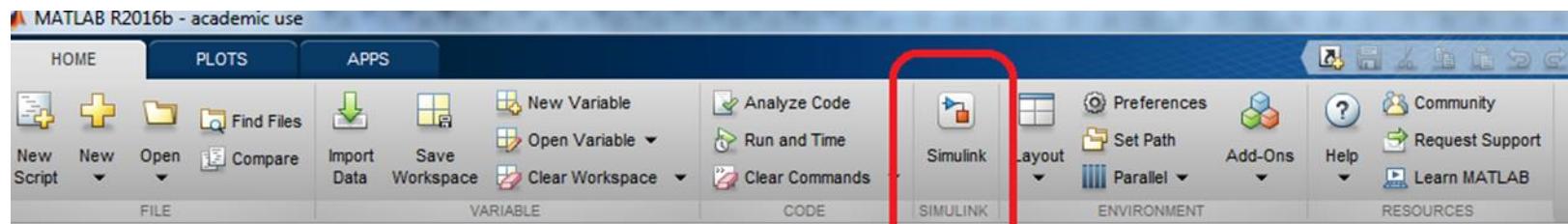
- a) Entorno Simulink.
  - b) Bloques. Modelos simples.
  - c) Entradas – salidas – puertos.
  - d) Operadores.
  - e) Condicionales: if-else
  - f) Representando datos
  - g) Ayuda

# 1. Introducción

- Es un entorno de programación gráfica sobre MATLAB.
- Muy utilizado como simulador de modelos
- Componentes. Algoritmos. Bloques.
- Facilidad de uso: curva de aprendizaje más suave que MATLAB.
- Muy útil para el control de hardware.
- Ayuda mucho conocer MATLAB

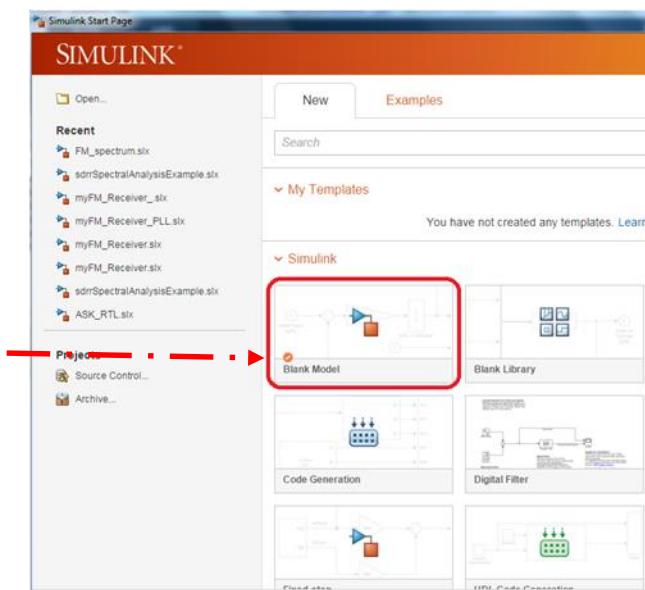


## 2. Entorno Simulink



Arrancamos simulink

Modelo en blanco  
(blank model)

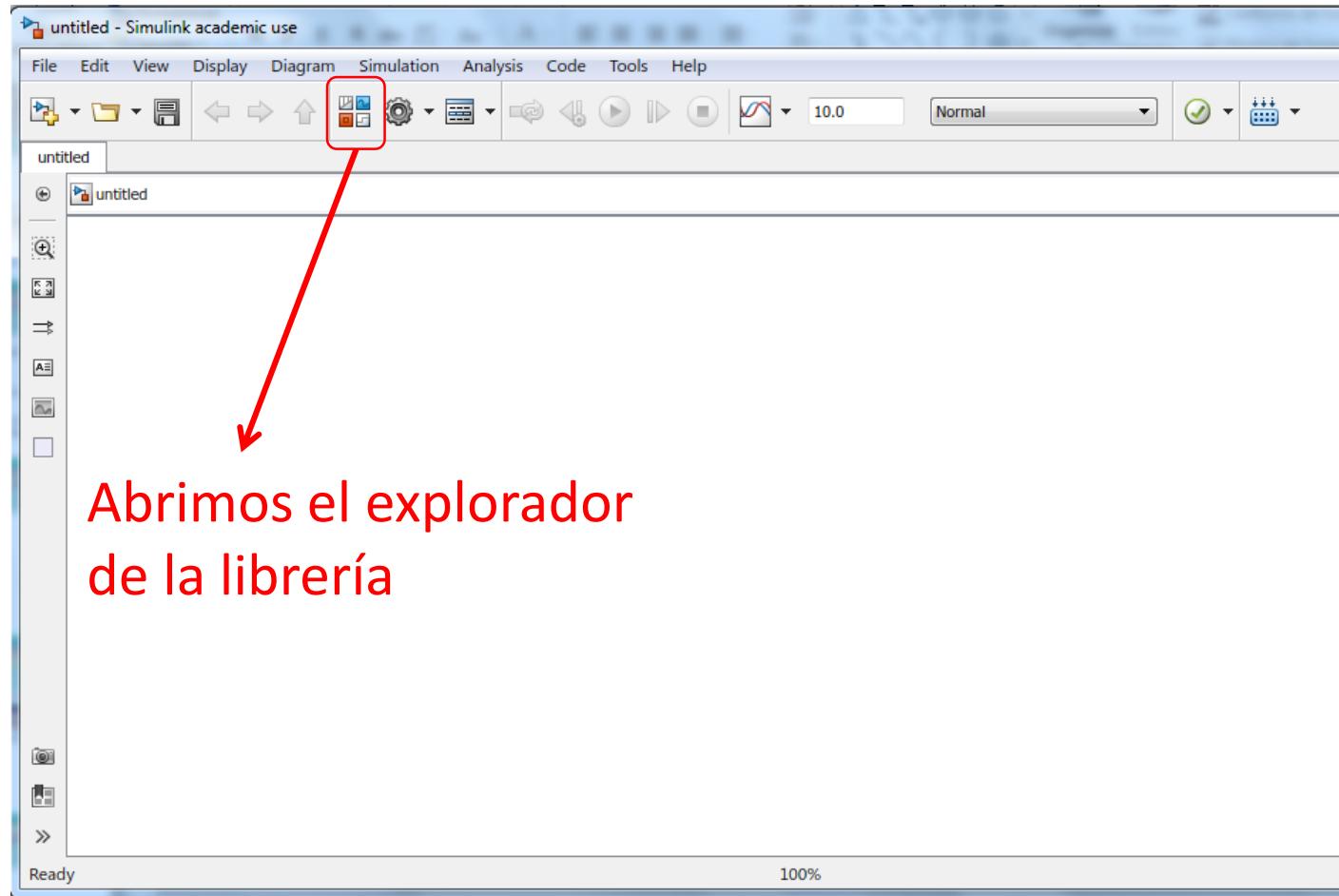


## 2. Entorno Simulink

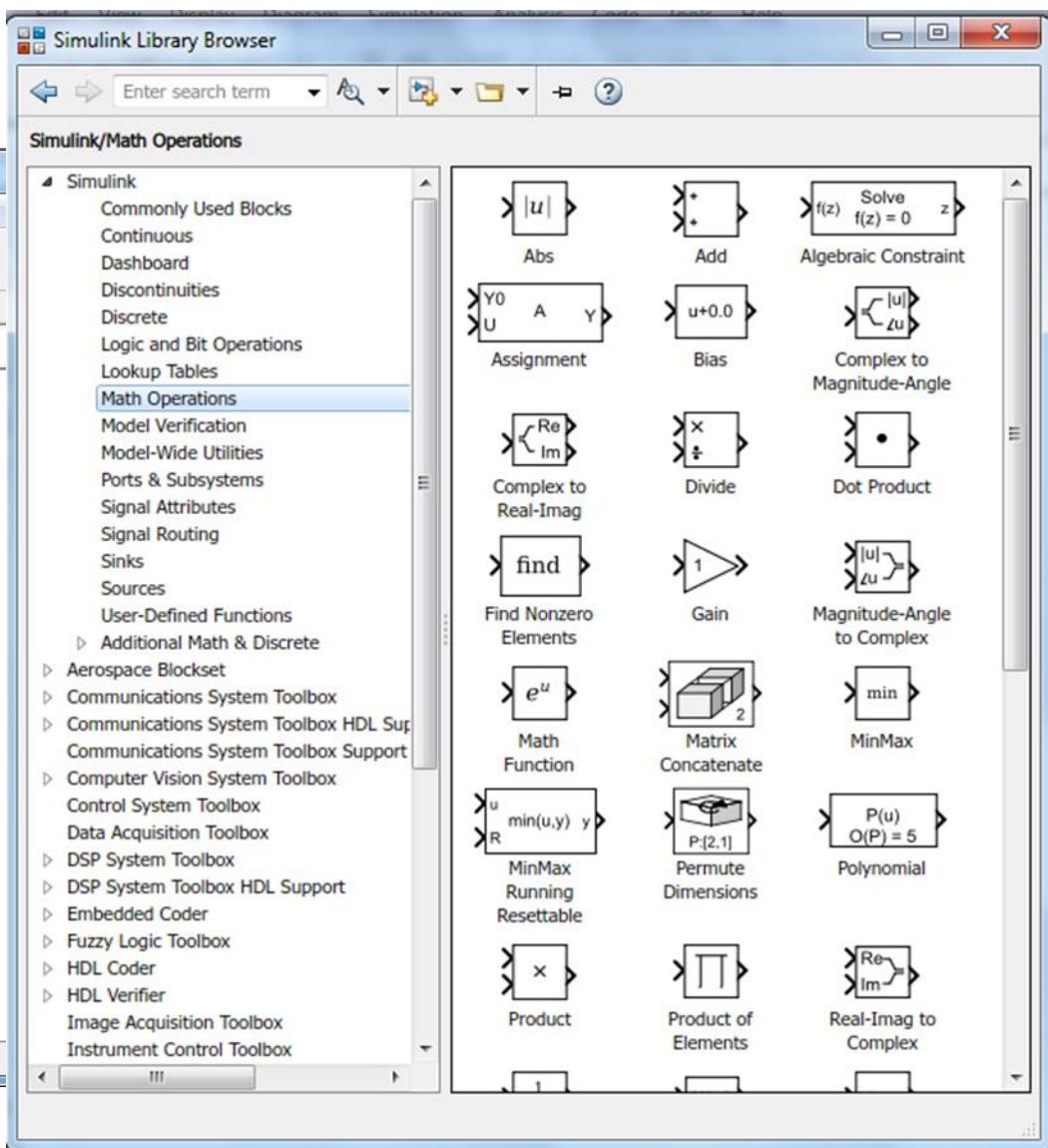


- 1 Arranca la simulación
- 2 Avanza/retrocede un paso de simulación
- 3 Tiempo máximo de simulación (puede ser infinito)
- 4 “Deploy to hardware” → carga el modelo en el HW Arduino, etc.
- 5 Librería de bloques
- 6 Opciones

# 2. Entorno Simulink

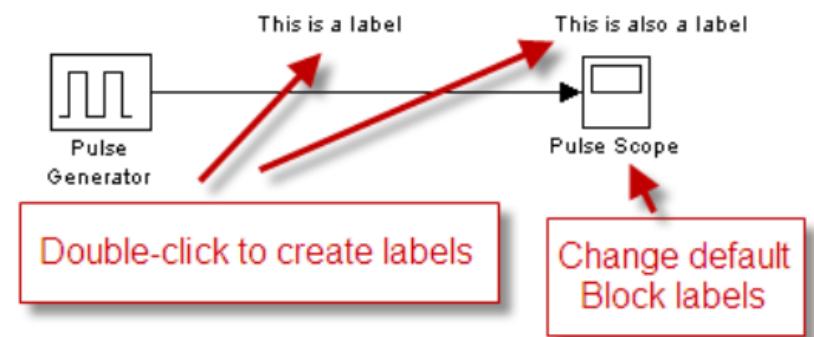


Abrimos el explorador de la librería



### 3. Bloques predefinidos

- Continuos.
  - Matemáticos.
  - ‘Enrutado’.
  - Fuentes (sources).
  - Sumideros (sink).
  - ...
- Podemos insertar comentarios y etiquetas (doble-click)
  - Hay que conectar todos los elementos.
  - Debe mantenerse la coherencia de los tipos y formatos de datos.

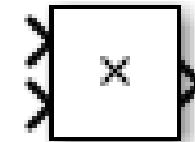


### 3. Bloques predefinidos. Primeros pasos

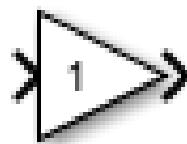
Sum



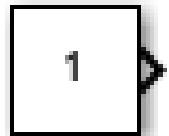
Product



Gain

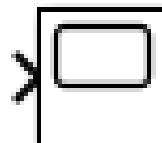


Constant



Terminator

Scope  
(Osciloscopio)



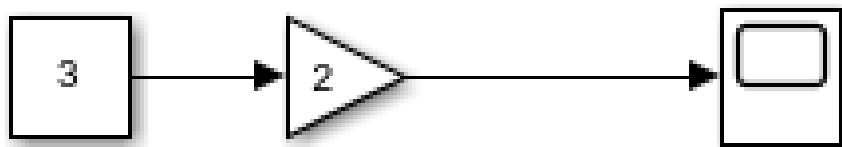
## Ejercicio 3.1

- Diseñe un modelo Simulink que, partiendo de una constante (por ejemplo 3), la multiplique por otra (por ejemplo, 2) y la represente en una gráfica.



# Ejercicio 3.1

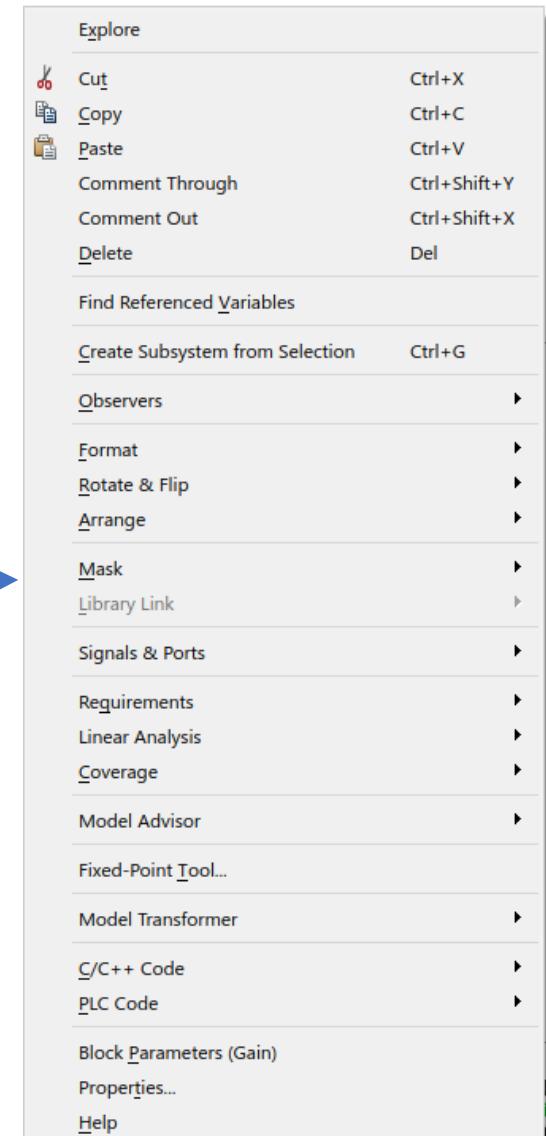
- Diseñe un modelo Simulink que, partiendo de una constante (por ejemplo 3), la multiplique por otra (por ejemplo, 2) y la represente en una gráfica.



# 3. Bloques predefinidos. Primeros pasos

- Sobre los bloques podemos realizar diversas operaciones y customizarlos
- #puertos
- Tipos
- Copia/pega/corta
- Comentar
- Parámetros del bloque
- Ayuda
- ...

Botón derecho



## Ejercicio 3.2

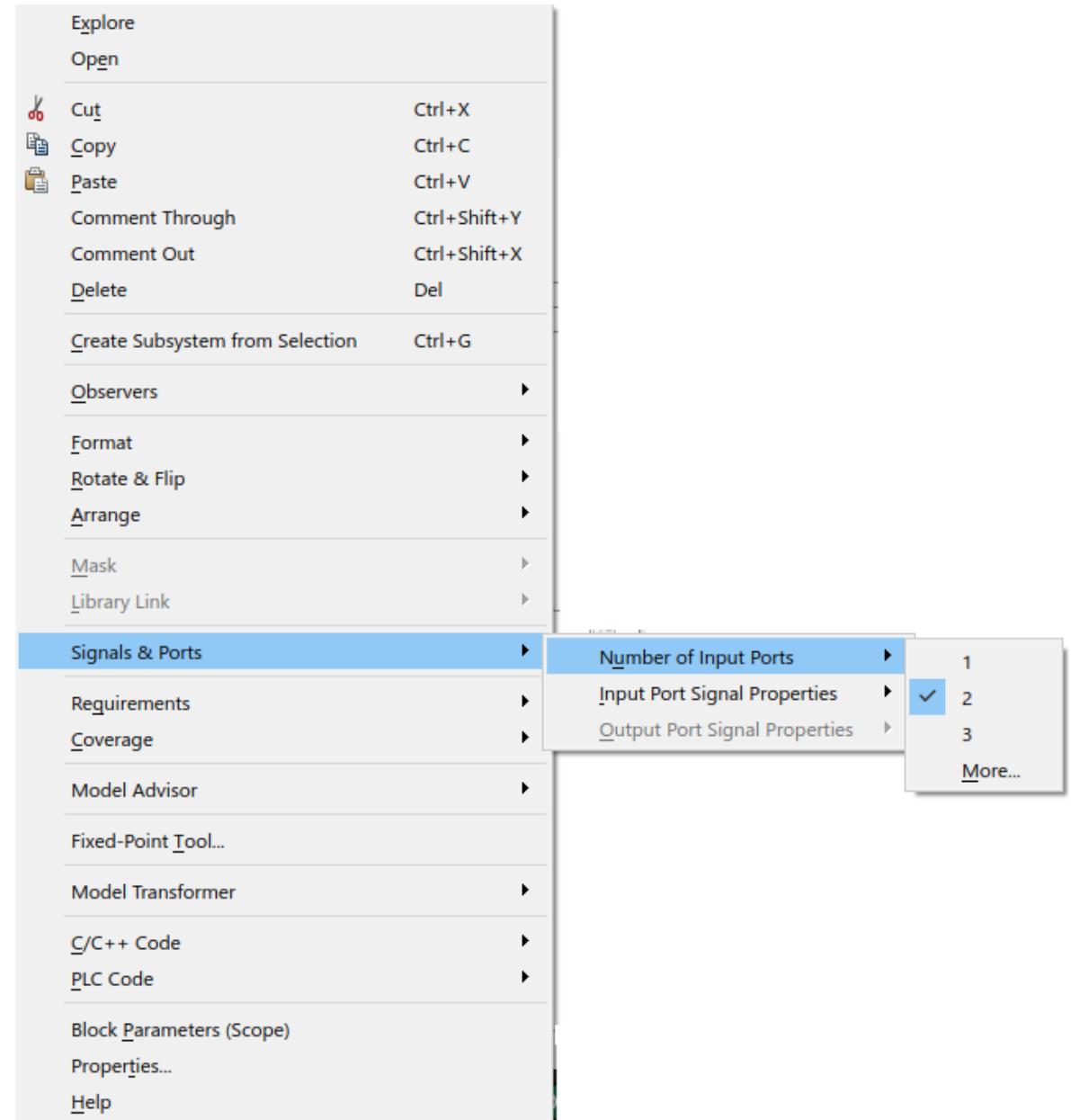
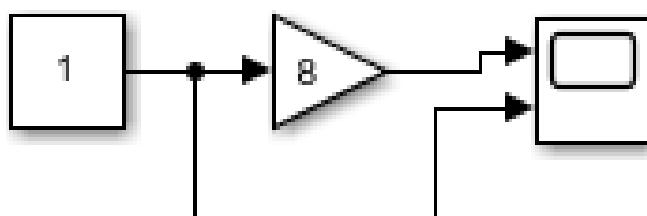
- Diseñe un modelo Simulink que, partiendo de una constante (por ejemplo 3), la multiplique por otra (por ejemplo, 2) y la represente en una gráfica junto con la constante original.



# Ejercicio 3.2



- Diseñe un modelo Simulink que, partiendo de una constante (por ejemplo 3), la multiplique por otra (por ejemplo, 2) y la represente en una gráfica junto con la constante original.



### 3. Bloques predefinidos. Primeros pasos

- Suma y resta se definen sobre el mismo bloque
- Producto y división se definen sobre el mismo bloque
- Productos matriciales, productos elemento a elemento
- Como no puede quedar ningún puerto sin conectar, usamos terminadores si es necesario
- Se pueden definir todo tipo de fuentes (*sources*) y “sumideros” (*sinks*) para nuestras señales

## Ejercicio 3.3

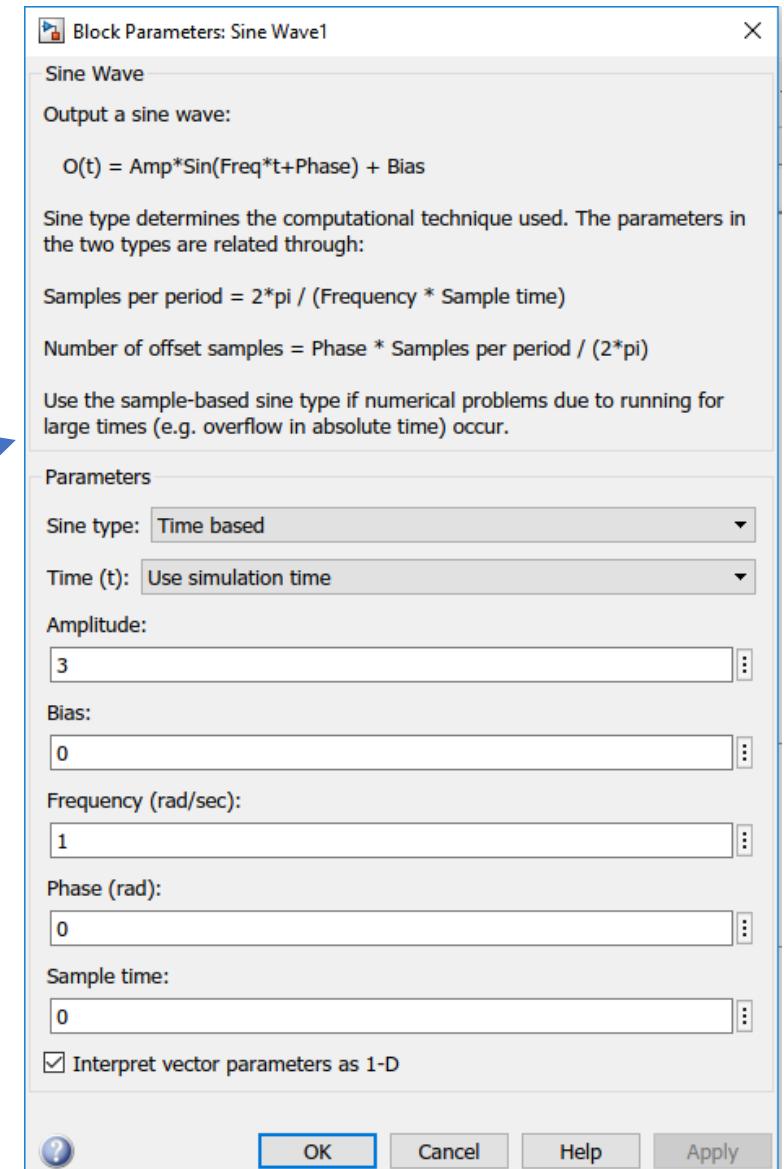
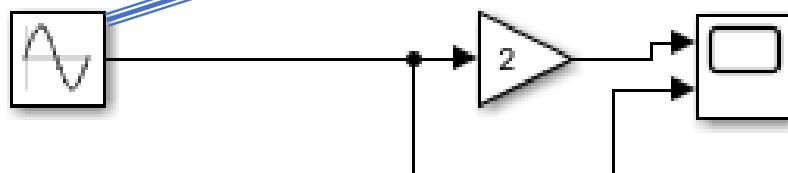
- Sustituya la constante del ejercicio anterior por una señal senoidal de pico 3, multiplíquela por la misma constante y represente ambas señales en el dominio del tiempo
- Ayuda: use el bloque “*sine wave*”



# Ejercicio 3.3



- Sustituya la constante del ejercicio anterior por una señal senoidal de pico 3, multiplíquela por la misma constante y represente ambas señales en el dominio del tiempo



# Ejercicio 3.4

- Genere dos señales senoidales de acuerdo a los datos de la tabla siguiente. Represente la resta de ambas señales.

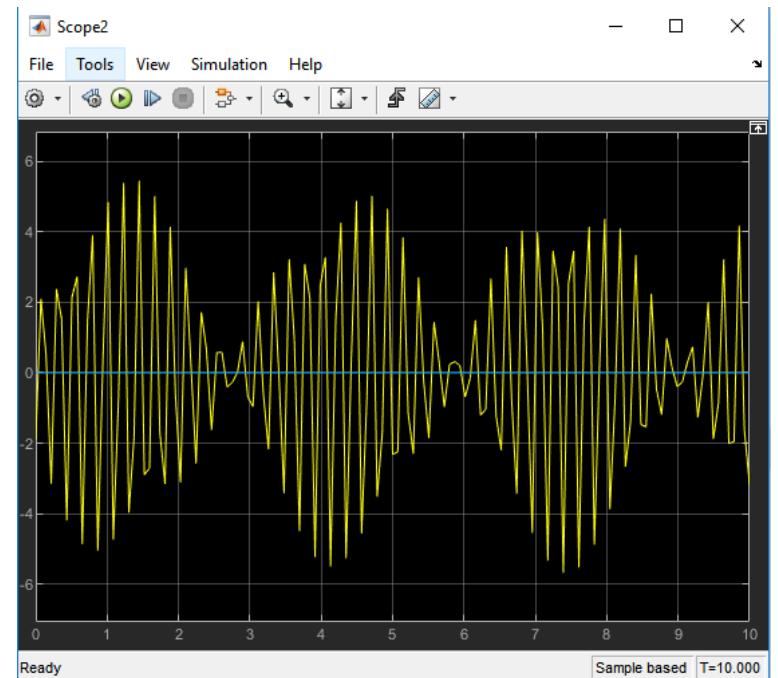
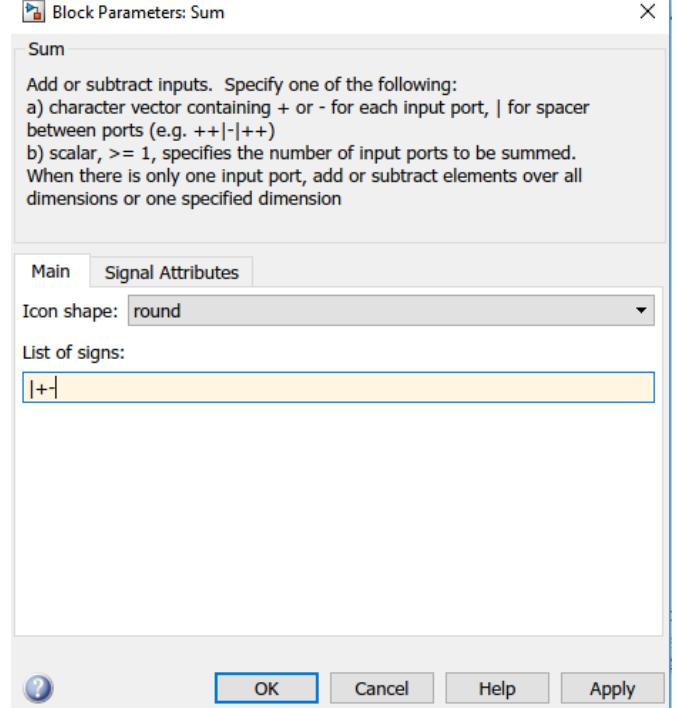
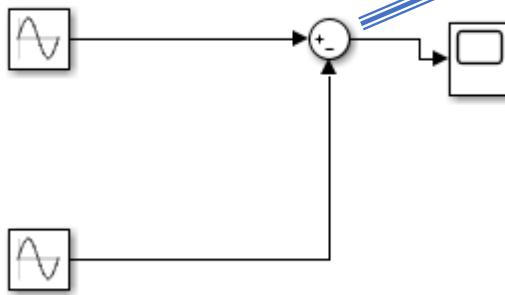
Señal	#1	#2
Amplitud	3 V	2.7 V
Frecuencia	27 Hz	29 Hz
Fase	0 deg	45 deg



# Ejercicio 3.4



- Genere dos señales senoidales de acuerdo a los datos de la tabla siguiente. Represente la resta de ambas señales.



### 3. Bloques predefinidos: sinks & sources



Display



Floating Scope



Out Bus Element



Out1



Scope



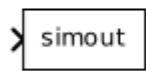
Stop Simulation



Terminator



To File



To Workspace



XY Graph

Sinks

Band-Limited  
White Noise

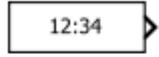
Chirp Signal



Clock



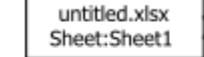
Constant

Counter  
Free-RunningCounter  
Limited

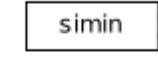
Digital Clock

Enumerated  
Constant

From File



From Spreadsheet

From  
Workspace

Ground



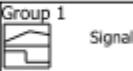
In Bus Element



In1

Pulse  
Generator

Ramp

Random  
NumberRepeating  
SequenceRepeating  
Sequence  
InterpolatedRepeating  
Sequence  
Stair

Signal Builder



Signal Editor

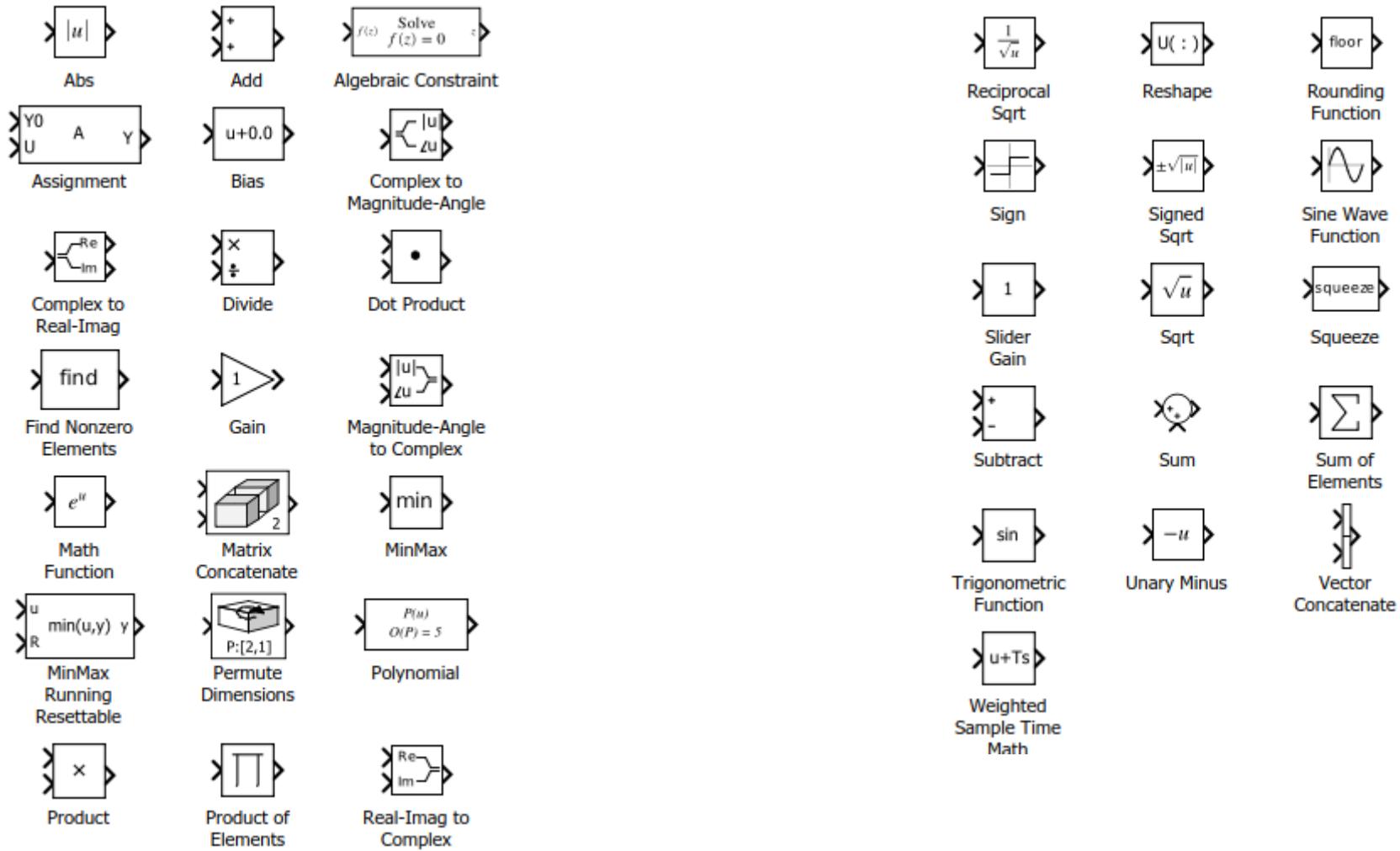


Step

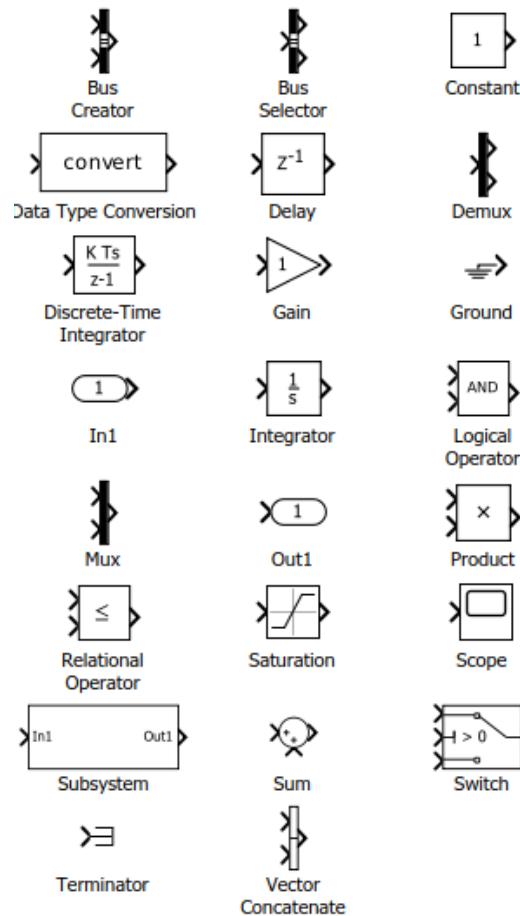
Uniform Random  
NumberWaveform  
Generator

Sources

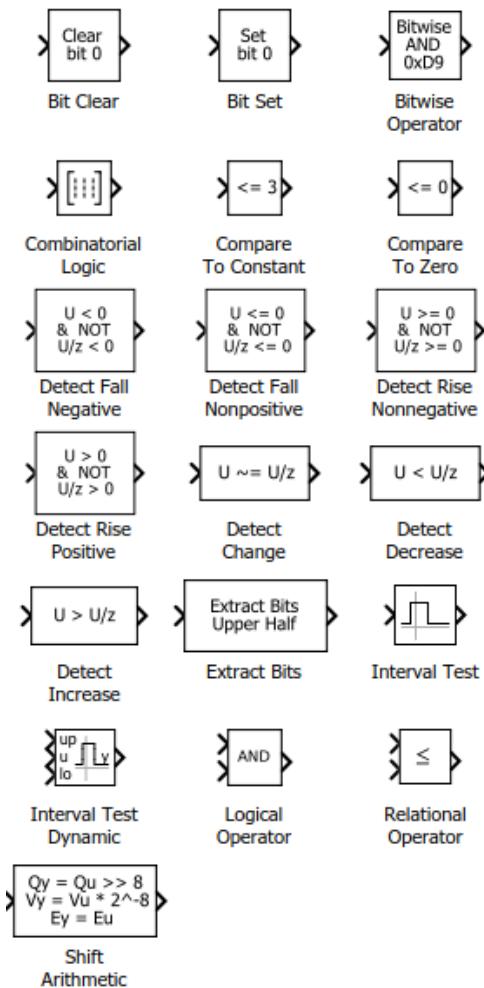
### 3. Bloques predefinidos: operaciones matemáticas



# 3. Más bloques predefinidos



Los más comunes

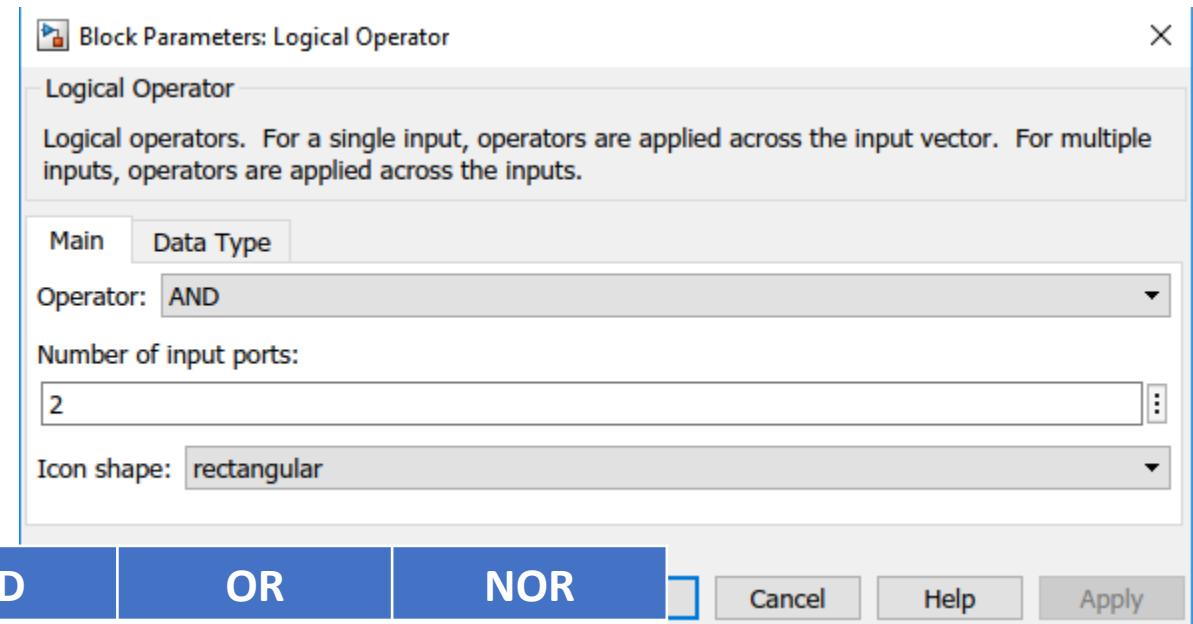
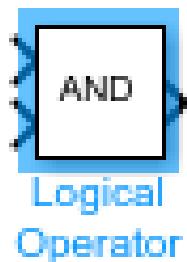


Bloques lógicos

# 4. Operadores lógicos

- En el bloque “AND” podemos elegir cualquier otra función lógica:

- OR
- NOR
- AND
- NAND
- XOR
- NXOR
- NOT

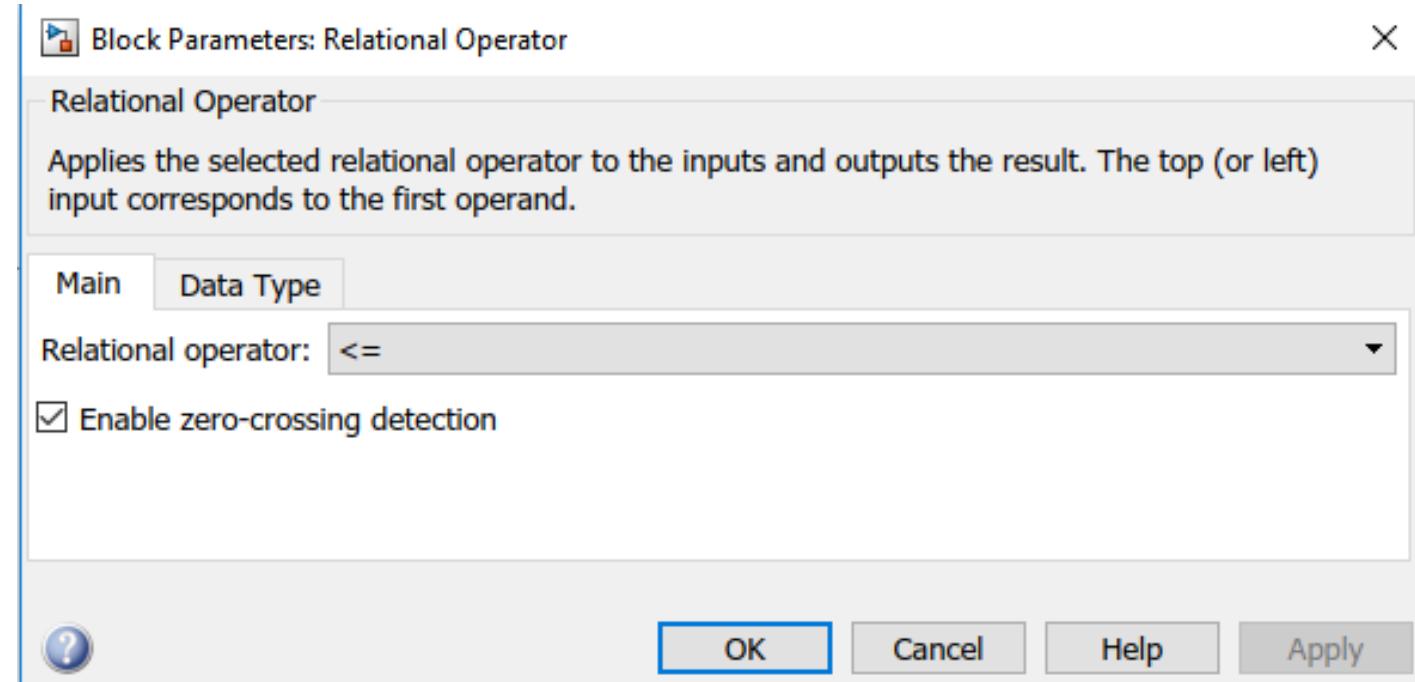
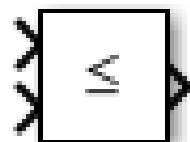


A	B	AND	OR	NOR
0	0	0	0	1
0	1	0	1	0
1	0	0	1	0
1	1	1	1	0

# 4. Operadores relacionales

- En el bloque “relational operator” podemos elegir cualquier otra función lógica:

- $\geq$
- $>$
- $\leq$
- $<$
- $\sim=$
- $=\!=$
- isNaN



# 5. Funciones if - else

Se utiliza para la ejecución condicional.

Sintaxis MATLAB.

if expresion

    código...

elseif expresion

    código...

else

    código...

end

Ejemplo:

a=0;

if g > 4

    a=1;

else

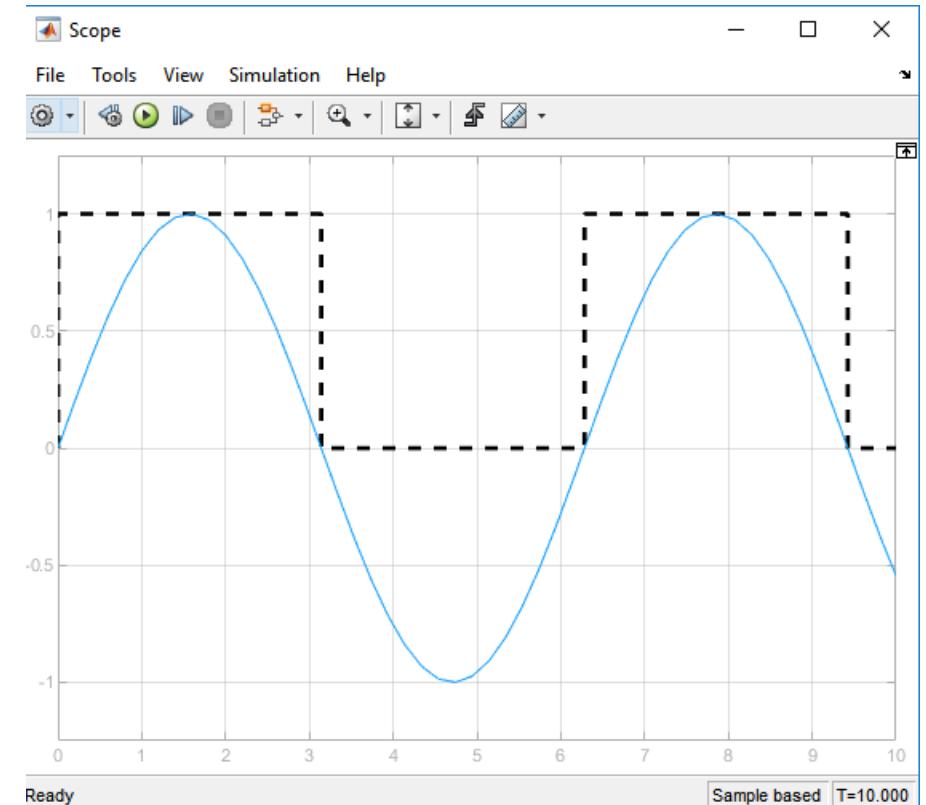
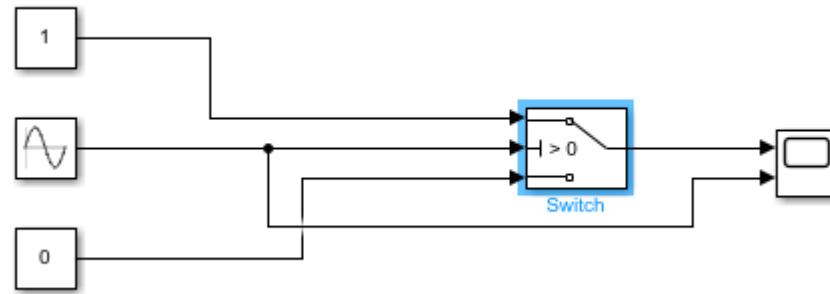
    a=0;

end

# 5. Funciones if - else

Ejemplo:

```
a=0;  
if g > 4  
    a=1;  
else  
    a=0;  
end
```



# Ejercicio 5.1

- Genere un modelo de simulink que conmute la salida entre dos posibles entradas de acuerdo a la siguiente expresión:
- $y = \begin{cases} 1, & \text{si } x_1 > \eta_1 \text{ AND } x_2 < \eta_2 \\ 0, & \text{si no} \end{cases}$

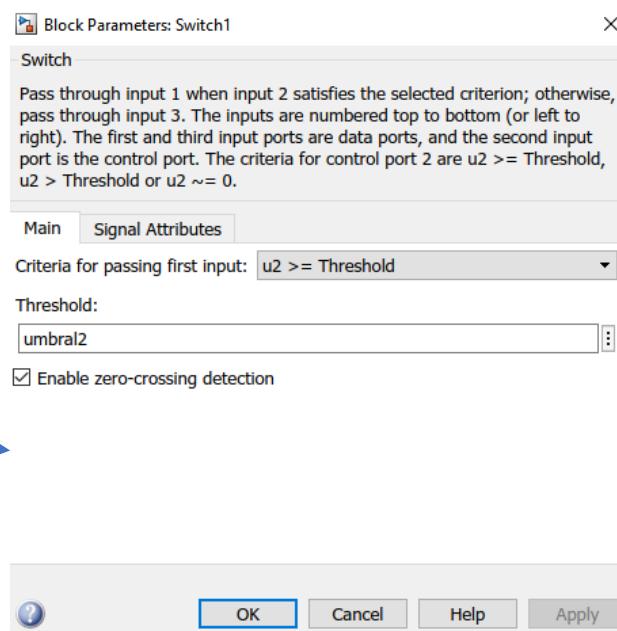
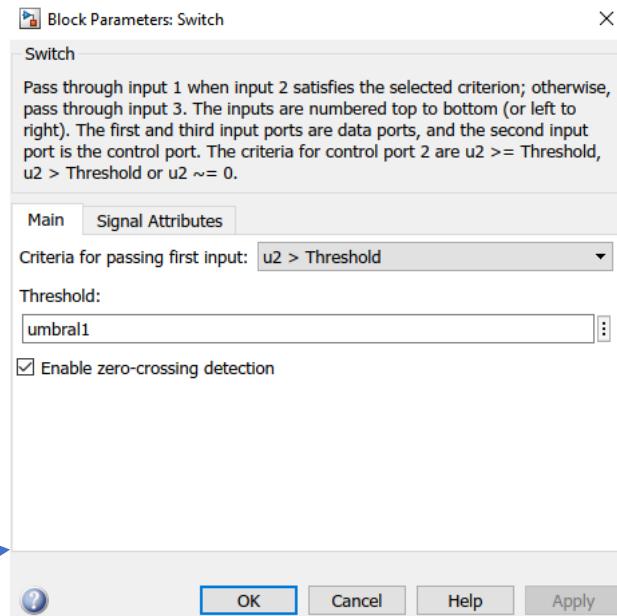
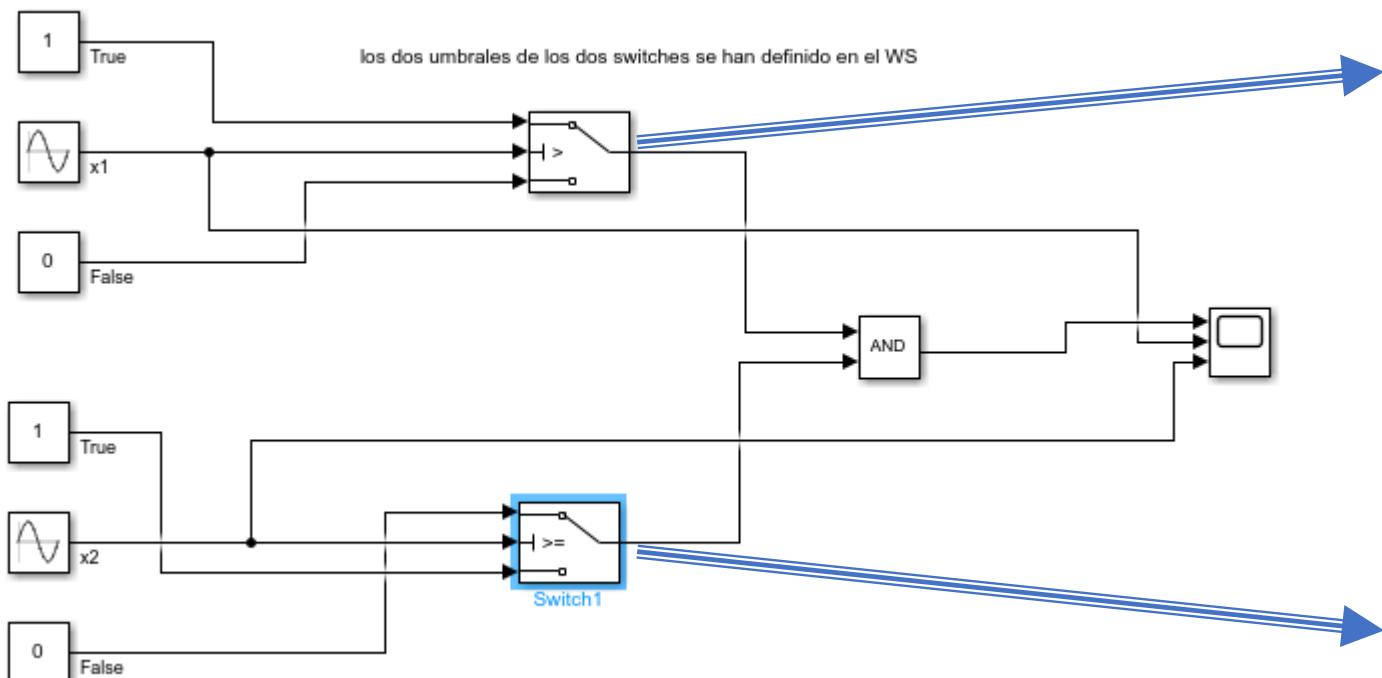
$x_1, x_2$  son sinusoides con diferente frecuencia y fase.

$\eta_1$  y  $\eta_2$  son constantes

No olvidar etiquetar las variables, y graficar la salida

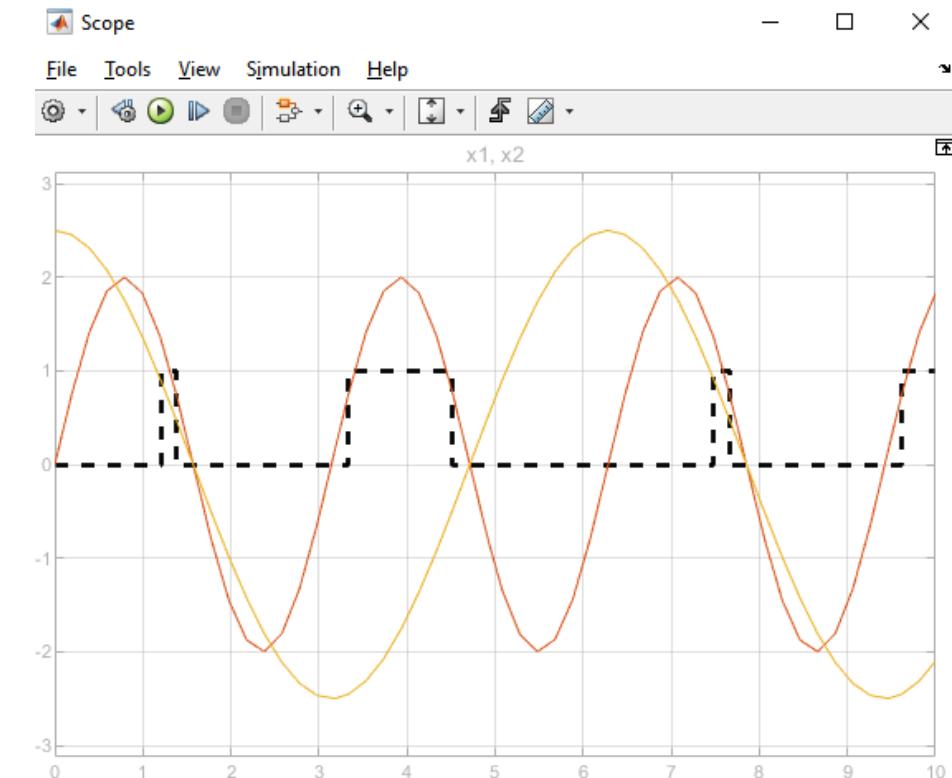


# Ejercicio 5.1

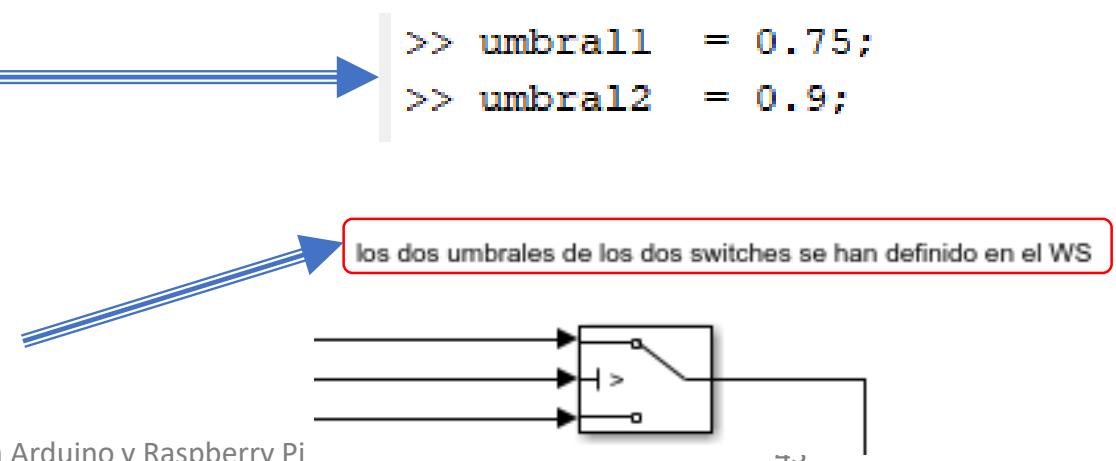


# Comentarios al ejercicio 5.1

- El ejercicio se podría haber completado con comparadores y no con switches
- Ver bloque 'compare to constant'
- Las variables umbral1 y umbral2 fueron definidas en el workspace de MATLAB
- Es conveniente comentar el 'código'

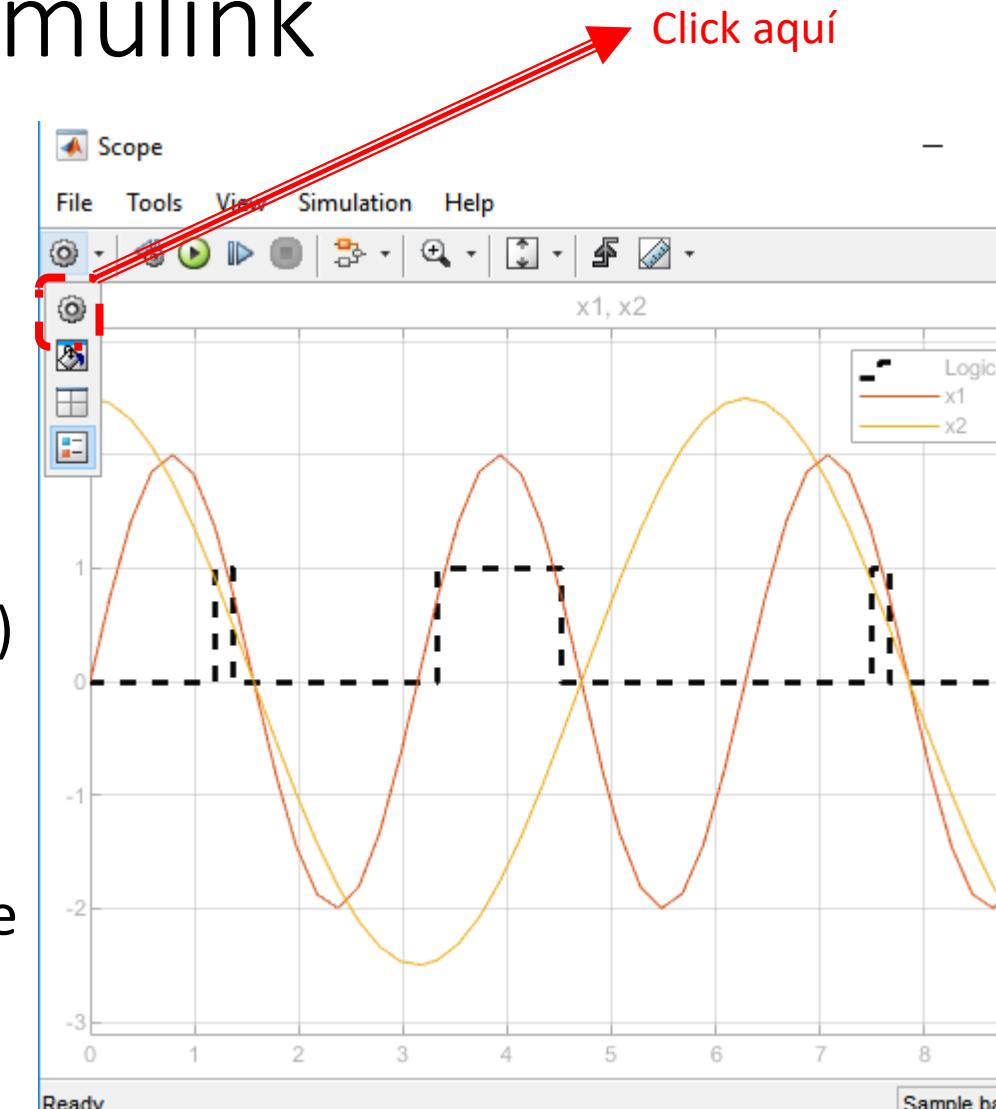


```
>> umbral1 = 0.75;  
>> umbral2 = 0.9;
```

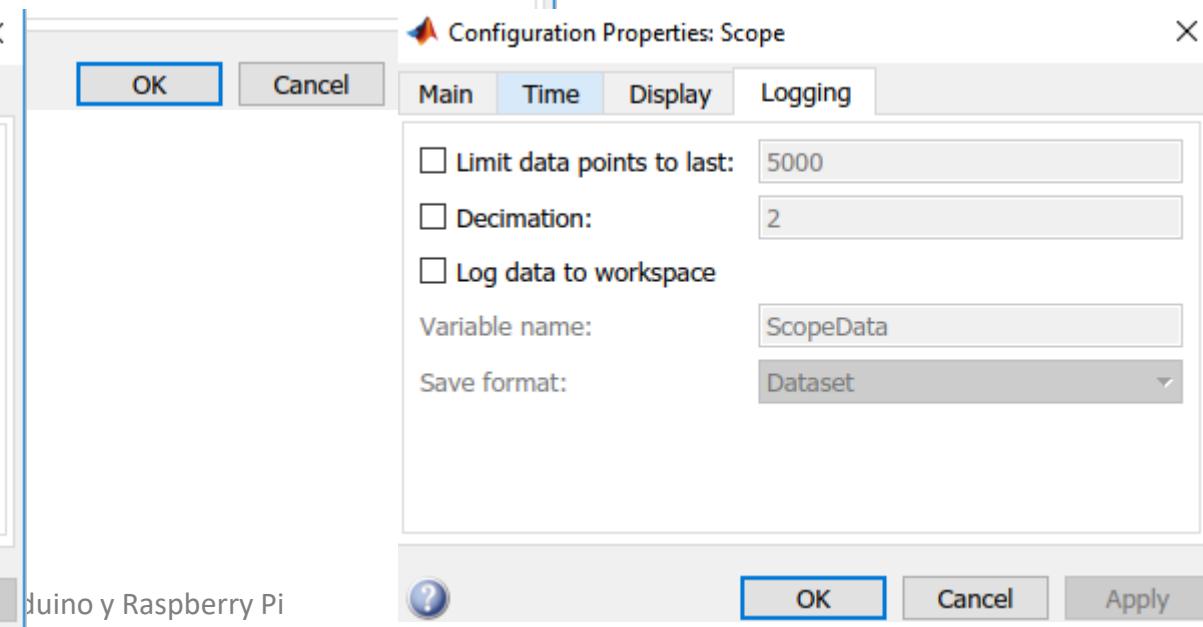
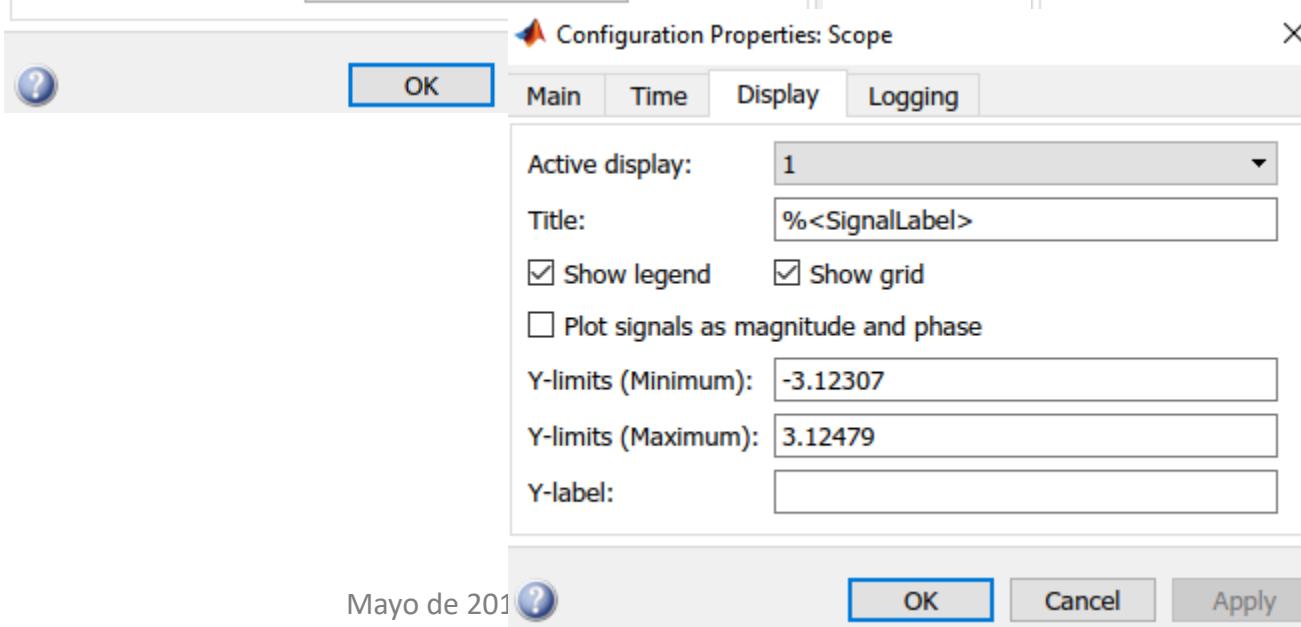
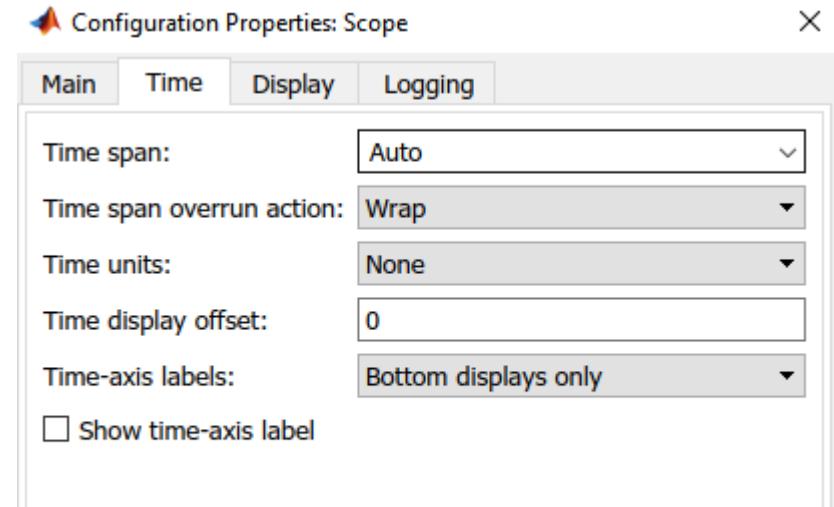
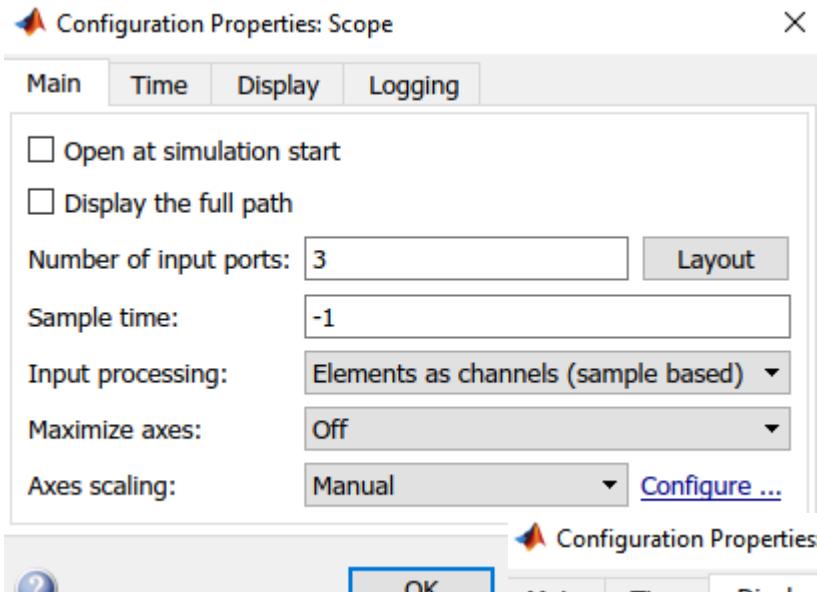


# 6. Representando datos en Simulink

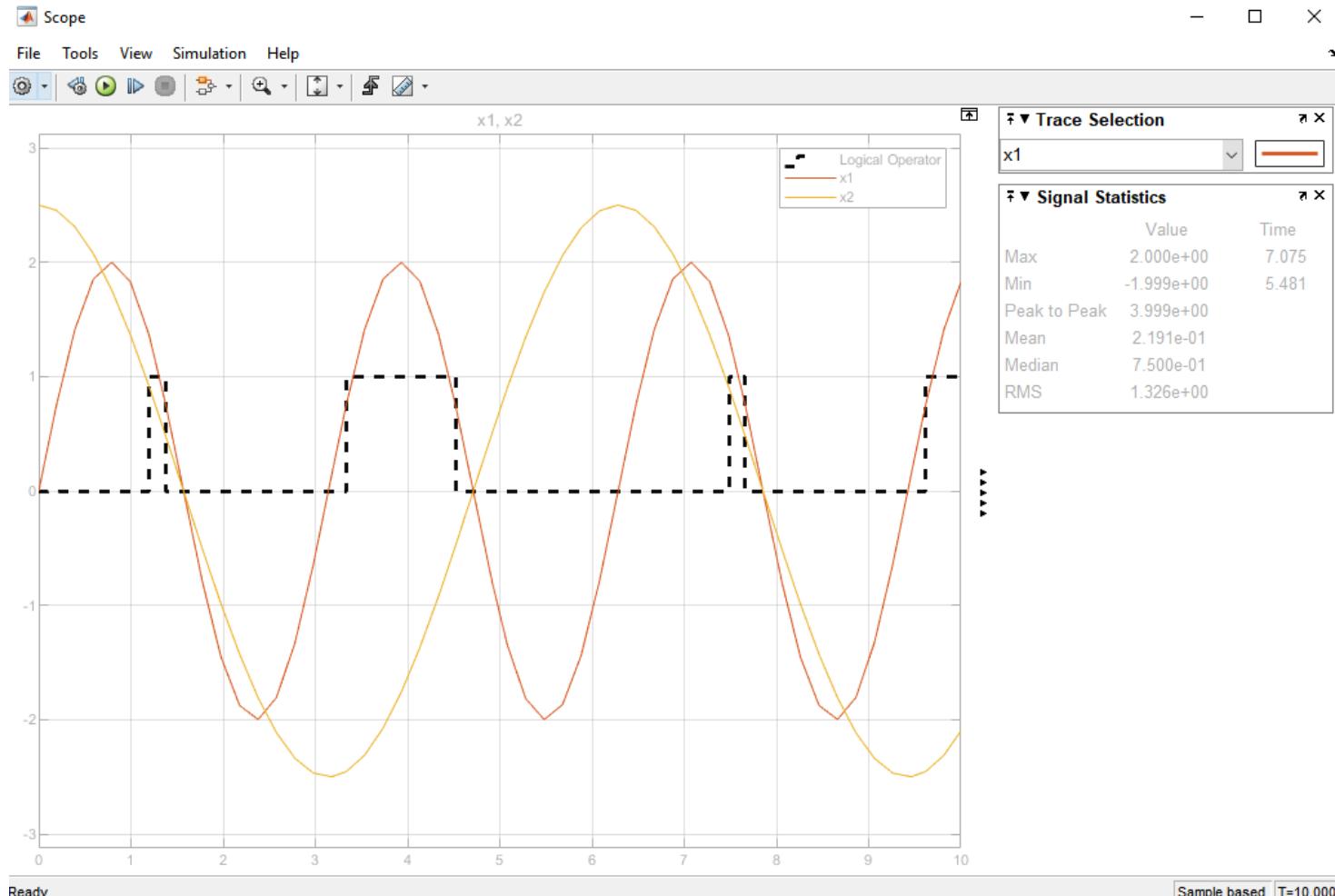
- Cuando graficamos datos, tenemos varias opciones aparte de scope
- En el menú ‘Configuration properties’ podemos:
  - Salvar los datos al WS de MATLAB
  - Cambiar opciones gráficas (colores, estilos, etc.)
  - Decidir si la gráfica se abrirá al empezar la simulación
  - Valores de los ejes
  - Tiempo de muestreo (-1 implica “heredar” el de las entradas)
  - ...



# 6. Representando datos en Simulink



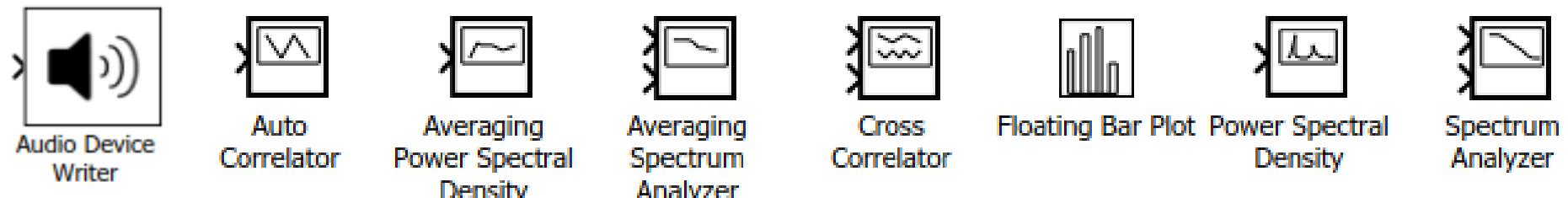
# 6. Representando datos en Simulink



- Zoom
- Exportar figuras a .fig (MATLAB) y a png, jpg, etc.
- Ver estadísticas de una señal, etc

# 6. Representando datos en Simulink

- Además de scope tenemos varias representaciones gráficas:
  - XY plot
  - Display (para mostrar un valor solamente)
- Y otros de propósito específico:
  - Analizador de espectro (spectrum analyzer)
  - Autocorrelador
  - Salida audio
  - ...



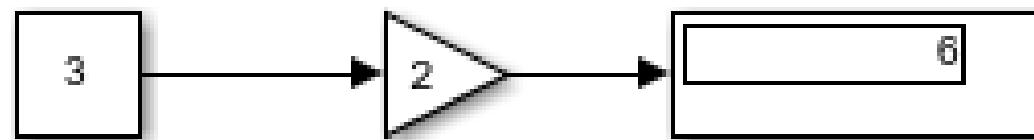
## Ejercicio 5.2

- Diseñe un modelo Simulink que, partiendo de una constante (por ejemplo 3), la multiplique por otra (por ejemplo, 2) y muestre su valor.



## Ejercicio 5.2

- Diseñe un modelo Simulink que, partiendo de una constante (por ejemplo 3), la multiplique por otra (por ejemplo, 2) y muestre su valor.

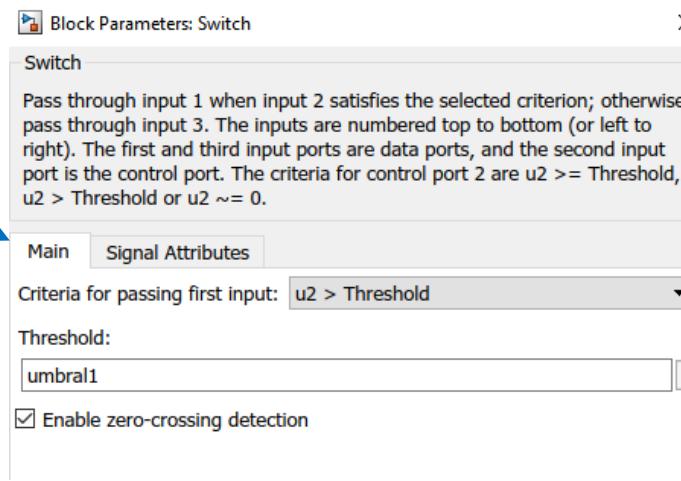
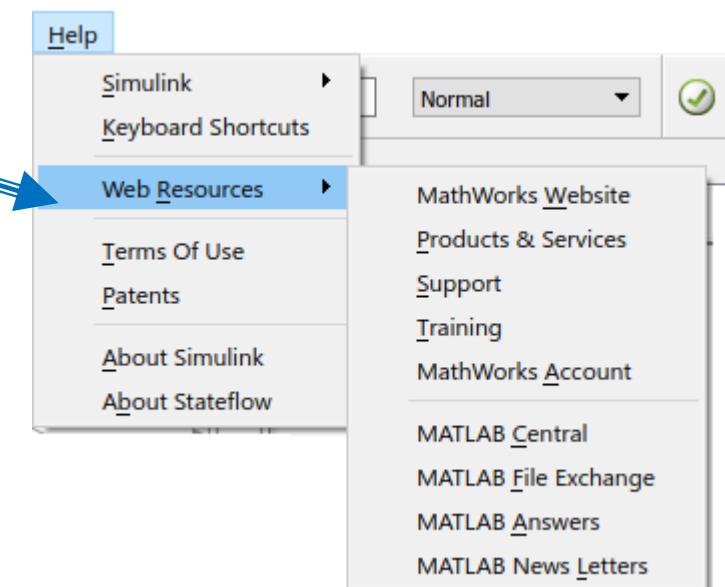
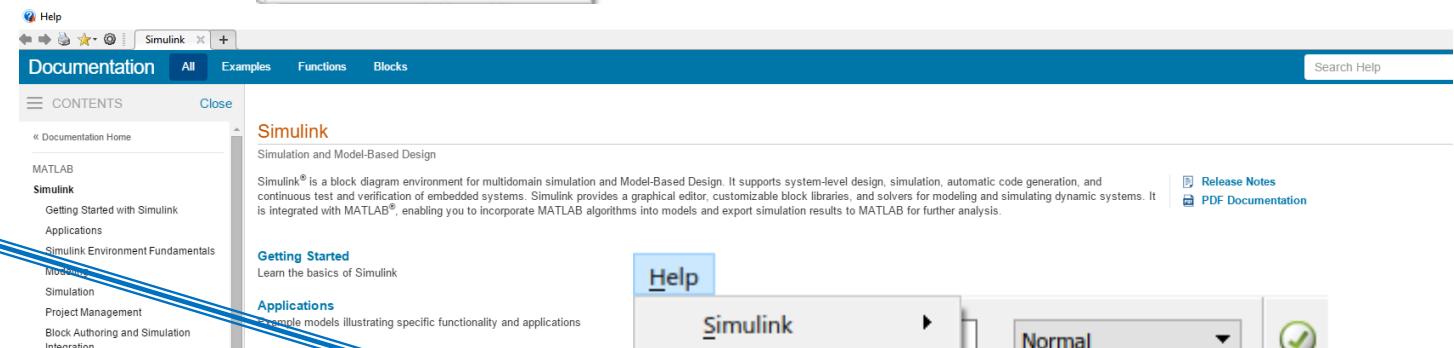
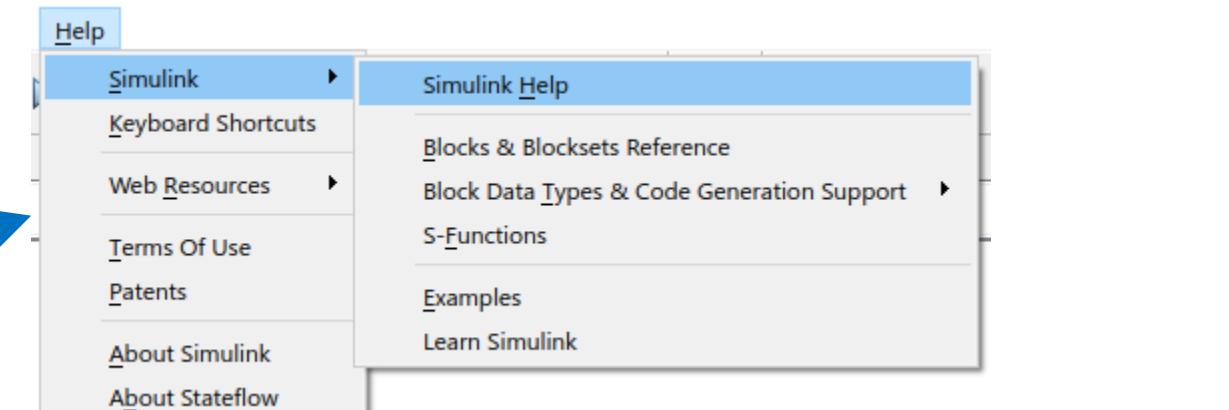


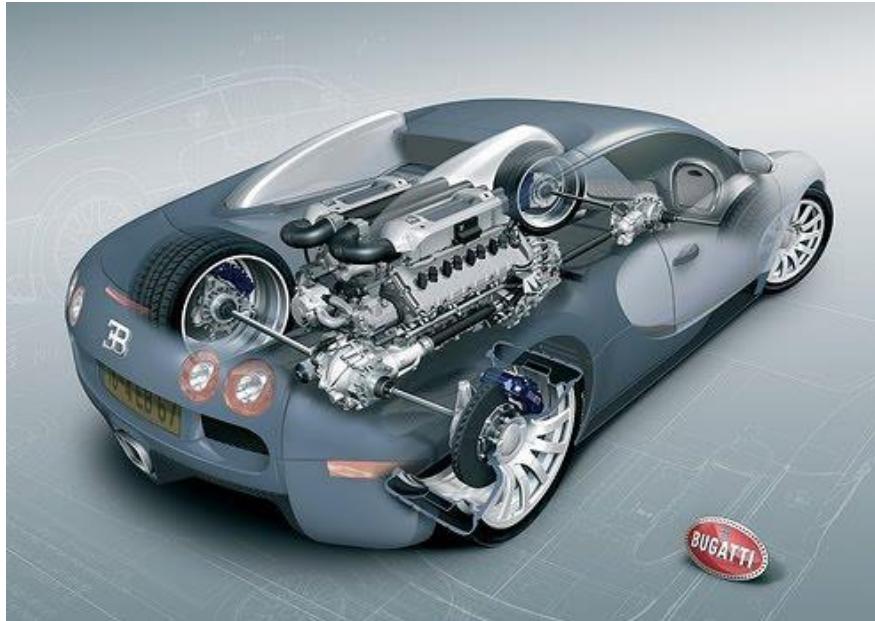
# 7. Ayuda

- Ayuda en línea:

- Simulink
- MATLAB Central
- MATLAB Answers
- Mail
- ...

- Block properties (doble click sobre bloque)





## Módulo 2. Simulink avanzado.

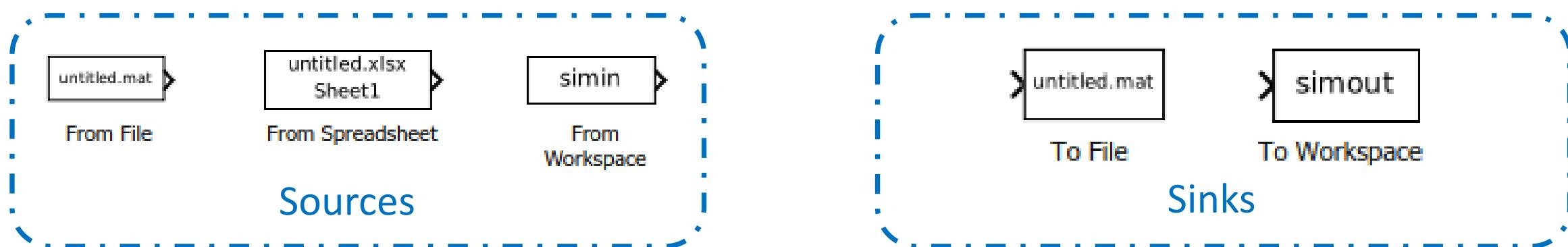
- a) Uso de variables y datos de MATLAB en Simulink (y viceversa).
- b) Código MATLAB en Simulink.
- c) Sistemas dinámicos discretos.
- d) Sistemas dinámicos continuos.
- e) Tiempo de simulación y 'step size'
- f) Configuración de MATLAB &Simulink para trabajar con Arduino y Raspberry Pi.

# 0. Interrelación MATLAB – Simulink

- Hasta ahora hemos trabajado como si fueran cosas independientes
- Grandes sinergias:
  - Uso de variables compartidas en ambos
  - Definición de bloques Simulink en base a funciones MATLAB
- Oportunidad: usar lo mejor de ambos mundos (opinión personal ☺)
  - Simulaciones y control de HW → mejor curva de aprendizaje en Simulink
  - Procesado de datos & representación gráfica → MATLAB
  - ... largo etcétera
- SW en MATLAB: scripts y funciones

# 1. Variables y datos compartidos

- El WS de MATLAB es accesible desde Simulink
- Podemos definir en Simulink:
  - Datos (memoria) con origen en el WS de MATLAB → sources
  - Datos (memoria) con destino en el WS de MATLAB → sinks
  - Salvar/cargar variables en/desde disco duro



# Ejercicio 1.1

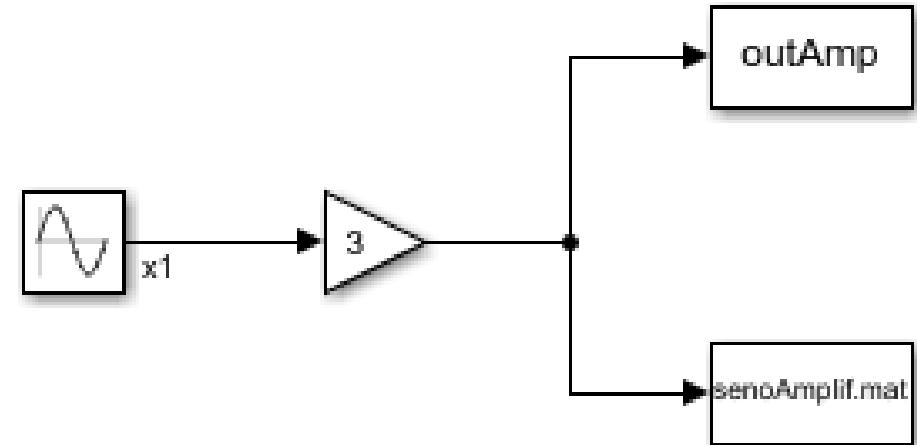
- Diseñe un modelo Simulink que, partiendo de una onda senoidal, la multiplique por una constante (por ejemplo, 3) y la represente en una gráfica junto con la onda original.
- Salve el resultado en un archivo llamado senoAmplif.mat y como una variable del workspace denominada outAmp.



# Ejercicio 1.1



- Diseñe un modelo Simulink que, partiendo de una onda senoidal, la multiplique por una constante (por ejemplo, 3) y la represente en una gráfica junto con la onda original.
- Salve el resultado en un archivo llamado senoAmplif.mat y como una variable del workspace denominada outAmp.



- Compruebe que se ha generado un archivo en el disco duro y que en la memoria (WS) aparece una nueva variable.

## 2. Scripts

Conjunto de operaciones en un archivo.

No tiene parámetros de entrada.

No tiene parámetros de salida.

Mismo workspace que ‘línea de comandos’.

Punto y coma para no mostrar resultado en línea de comandos.

```
%% script que resuelve el E08

mult3 = 3:3:(1000-1);
mult5 = 5:5:(1000-1);
mult15 = 15:15:(1000-1);
e08 = sum(mult3) + sum(mult5) - sum(mult15) % vale 233168
save E08.mat e08
```

## 2. Funciones

Conjunto de operaciones en un archivo.

Puede tener uno o varios parámetros de **entrada**.

Puede tener uno o varios parámetros de **salida**.

Dispone de un workspace interno, no visible desde otras funciones ni desde la ‘línea de comandos’.

```
function [ PdBm ] = wAdBm( p_W )
% wAdBm convierte potencia en W a dBm
%
% p_W debe ir en vatios
PdBm = 30 + 10*log10(p_W);
end
```

Función de una entrada y una salida

```
function [m,s] = stat(x)
%% esta funcion devuelve la media y la desviacion tipica de un vector
n = length(x);
media = sum(x)/n;
desvTip = sqrt(sum((x-media).^2/n));
end
```

Función de una entrada y dos salidas

## 2. Funciones

Se deben almacenar en un archivo llamado ‘nombreFuncion.m’.

Aplica la “regla del punto y coma”.

Tienen un workspace interno.

Function [salida1, salida2, ...] = nombreFuncion(entrada1, entrada2, ...)

```
-function [ PdBm ] = wAdBm( p_W )
% wAdBm convierte potencia en W a dBm
%
% p_W debe ir en vatios
PdBm = 30 + 10*log10(p_W);
end
```

Función de una entrada y una salida

```
-function [m,s] = stat(x)
%% esta funcion devuelve la media y la desviacion tipica de un vector
n = length(x);
media = sum(x)/n;
desvTip = sqrt(sum((x-media).^2/n));
end
```

Función de una entrada y dos salidas

## 2. Usando código MATLAB

- Es posible hacer uso de código MATLAB en simulink
- Usaremos el bloque MATLAB Function.
- Los puertos de entrada son los parámetros de entrada y los de salida, las salidas de la función.
- Se puede cargar/salvar variables en el workspace MATLAB (en cualquier modelo Simulink, no solo en los definidos por nosotros).
- Herramienta potente, pero riesgo de “reinvención de rueda” → mejor usar bloques y funciones predefinidas de MATLAB



MATLAB Function

# Ejercicio 2.1

E2.1. Escriba una función que calcule el MCM, el MCD de dos números enteros.

Esta función deberá recibir dos enteros y devolver el MCM y el MCD de ambos.

Use esa función en Simulink como un bloque.

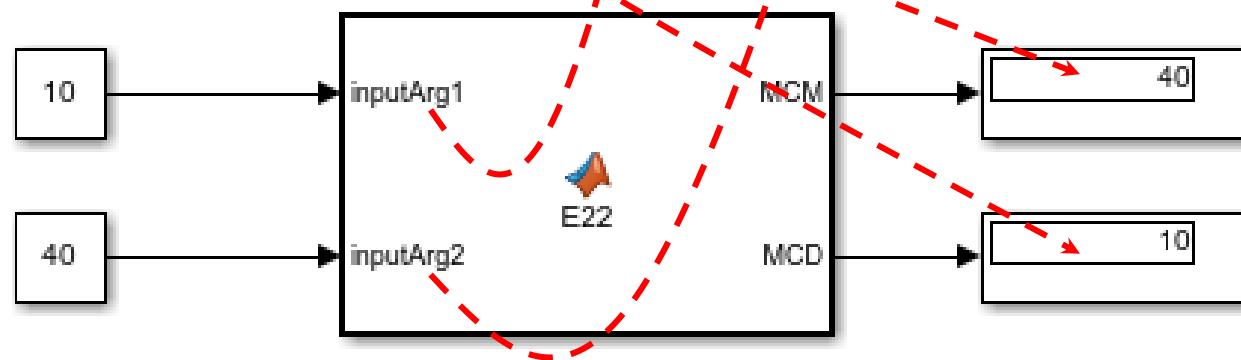
Nota: puede usar las funciones de MATLAB *lcd* y *gcm*



# Ejercicio 2.1



```
[function [MCM,MCD] = E22(inputArg1,inputArg2)
%función que calcule el MCM, el MCD de 2 números dados
    MCM = lcm(inputArg1,inputArg2);
    MCD = gcd(inputArg1,inputArg2);]
end
```



## Ejercicio 2.2

E2.2. Sobre el ejemplo anterior, modifique el modelo de forma que ahora tengamos una tercera salida (por ejemplo, el producto de las dos entradas)



## 2. Bonus: live scripts

Aparte de las funciones y scripts ‘normales’ también tenemos ‘live scripts’ y ‘live functions’.

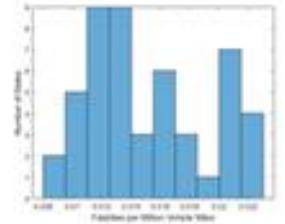
Extensión del archivo es \*.mlx.

Permite interacción ‘en caliente’.

### Distribution of Fatalities

We can use a bar chart to see the distribution of fatality rates among the states. There are 11 states that have a fatality rate greater than 0.02 per million vehicle miles.

```
histogram(rate,10)
xlabel('Fatalities per Million Vehicle Miles')
ylabel('Number of States')
```

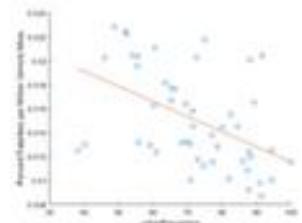


### Find Correlations in the Data

We can experiment with the data to see if any of the variables in the table are correlated with highway fatality rates. It appears that highway fatality rates are lower in states with a higher percentage urban population.

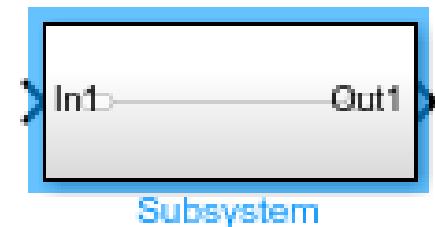
```
varName = 'urbanPopulation';
scatter(fatalities.(varName),rate)
xlabel(varName)
ylabel('Percent Fatalities per Million Vehicle Miles')

hold on
xmin = min(fatalities.(varName));
xmax = max(fatalities.(varName));
p = polyfit(fatalities.(varName),rate,1);
plot([xmin xmax], polyval(p,[xmin xmax]))
```



### 3. Encapsulado de sistemas

- Para simplificar el *layout* de nuestro modelo, es buena política crear subsistemas
- Se crea un bloque “subsistema” (subsystem)
  - Doble-click
  - Se copia y lega (o añade y conecta) el contenido
  - Importante definir bien los puertos de entrada y salida con In y Out, respectivamente



## Ejercicio 3.1

- Diseñe un subsistema que tenga dos entradas y dos salidas
- La primera salida será la suma de las dos entradas y la segunda salida será el producto de las dos entradas



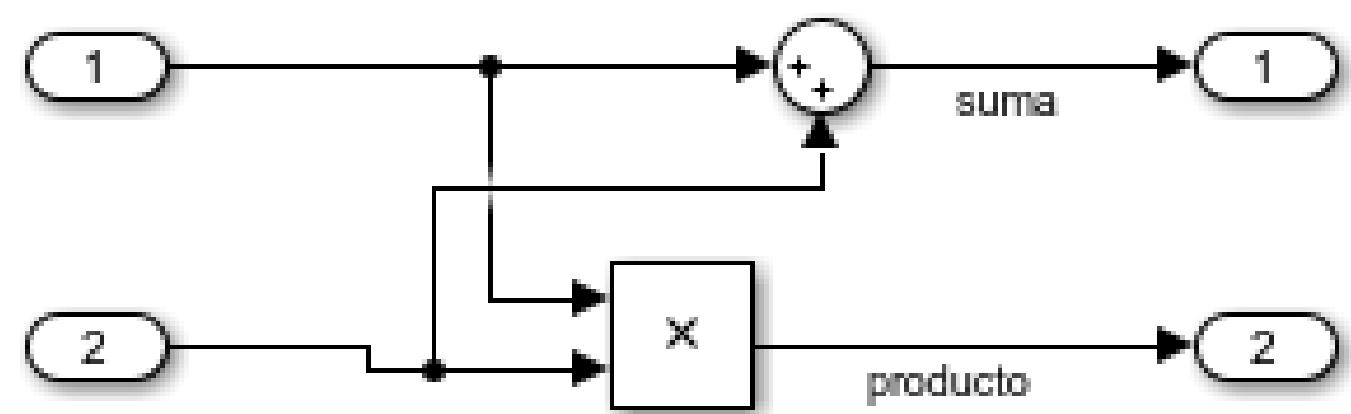
# Ejercicio 3.1



- Diseñe un subsistema que tenga dos entradas y dos salidas
- La primera salida será la suma de las dos entradas y la segunda salida será el producto de las dos entradas



Doble-click

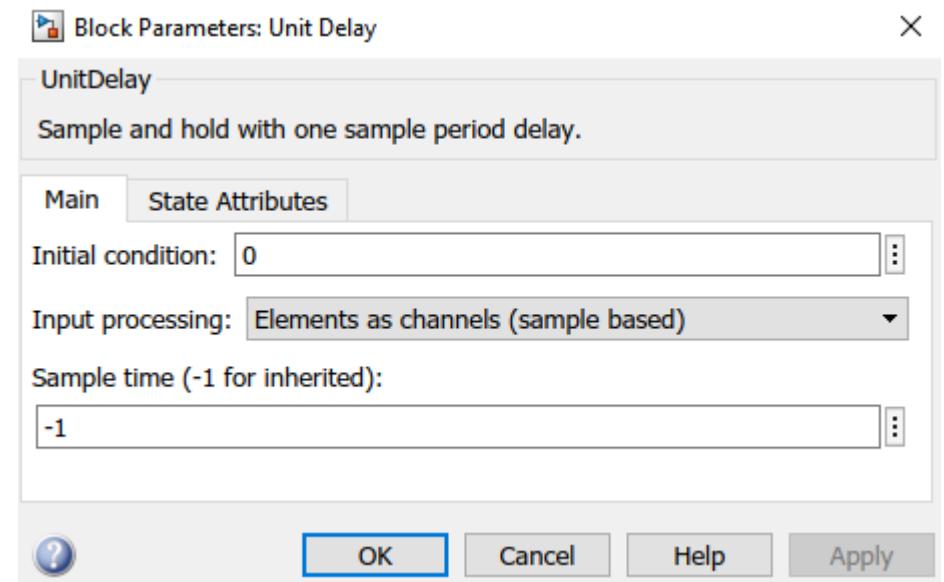
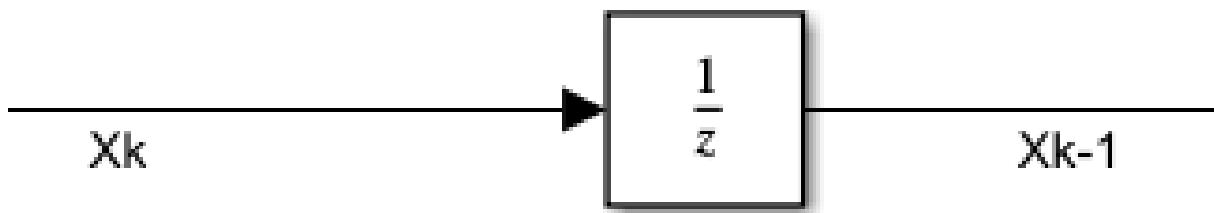


## 4. Sistemas dinámicos

- Sistema cuyo estado varía con el tiempo, en función de otras variables y de los estados previos
- Pueden ser
  - Discretos: solo se modela en instantes determinados (a intervalo fijo)
    - Sample time: concepto fundamental porque define el intervalo
  - Continuos: se modela todo el tiempo (sample time = 0)

# 4. Sistemas dinámicos discretos

- Bloque retardo unitario (unit delay)



- Permite indicar la condición inicial
- Sample time:
  - -1 para heredarlo de la entrada
  - Otros valor (especificar)

# Ejercicio 4.1

- Genere una onda senoidal (*sample time* = 0.5) y retrásela una unidad de tiempo. Represente ambas señales usando scope y mantenga el *sample time*.
- Pruebe a variar el *sample time* **en unit delay** a 0.25. Compare ambos resultados

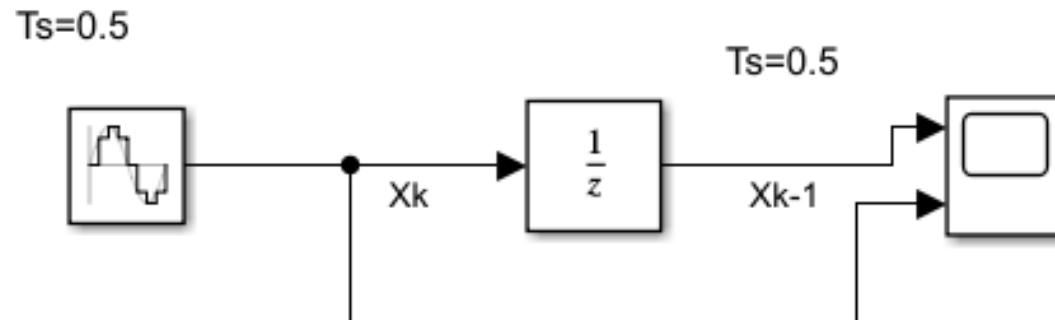


# Ejercicio 4.1

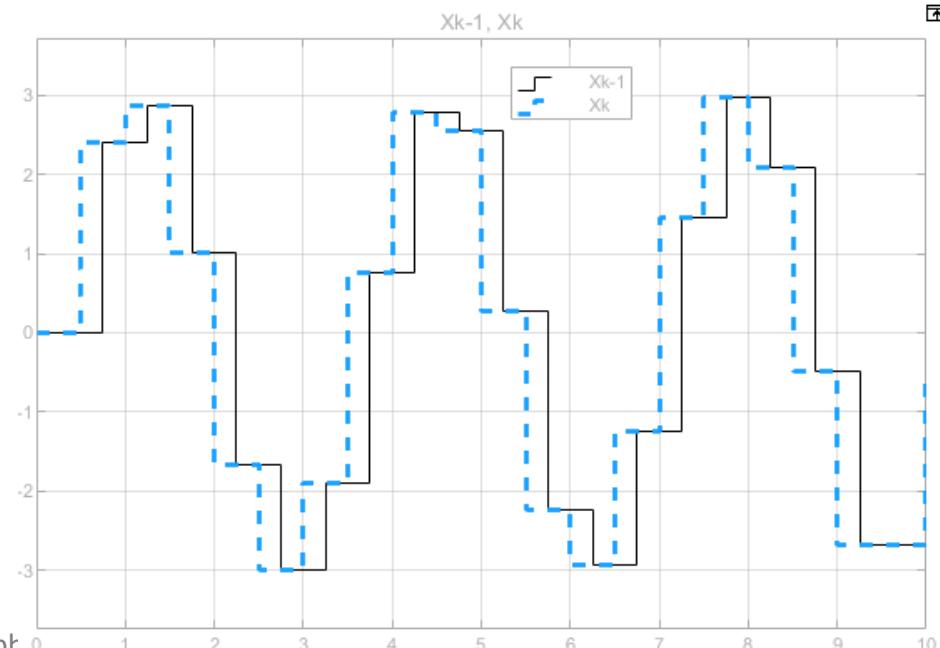
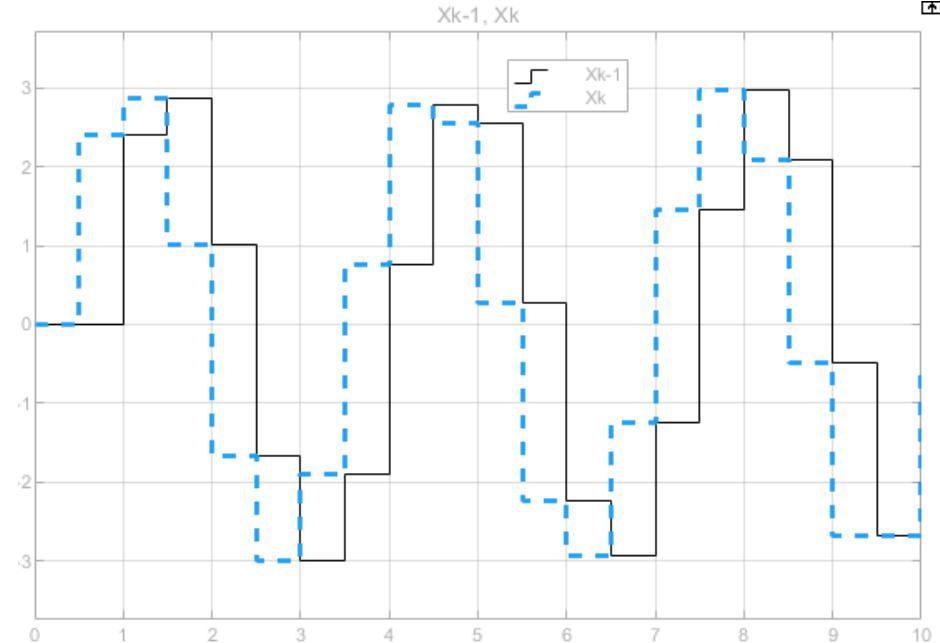


$T_s = -1$   
(0.5)

- Genere una onda senoidal (*sample time* = 0.5) y retrásela una unidad de tiempo. Represente ambas señales usando scope y mantenga el *sample time*.
- Pruebe a variar el *sample time* en *unit delay* a 0.25

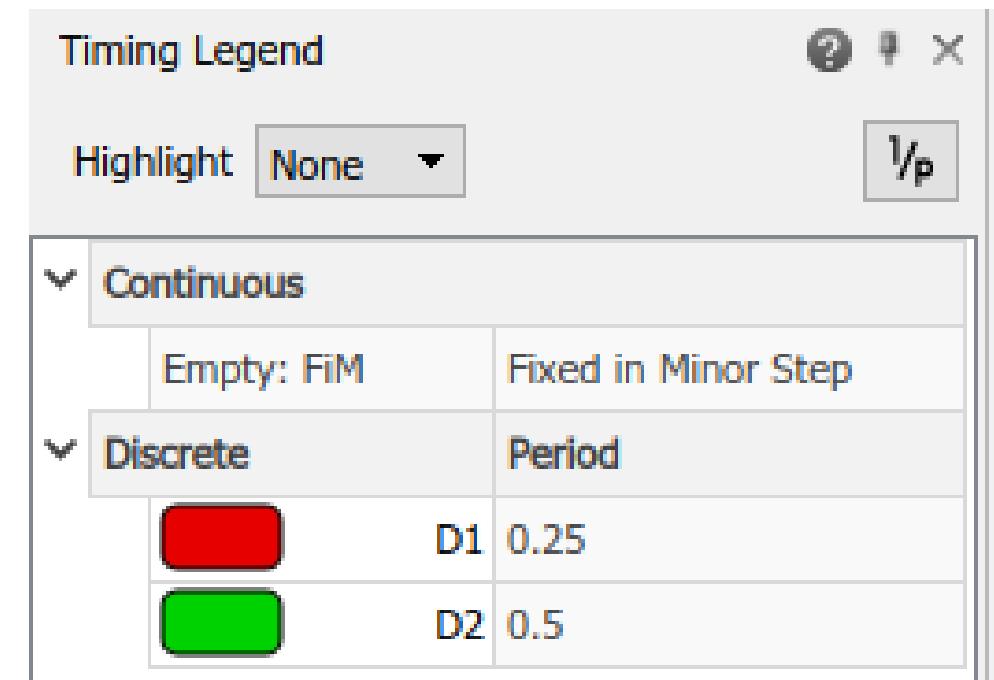
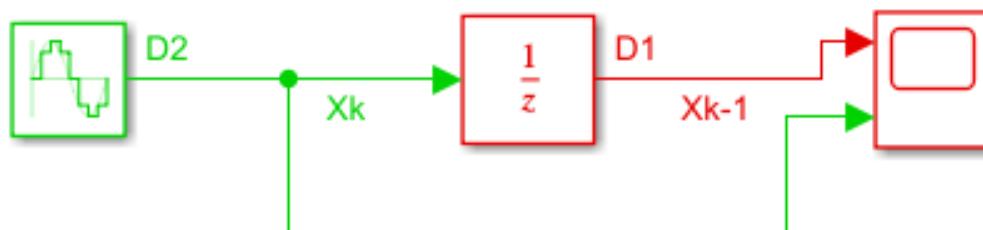


$T_s = 0.5$



# 4. Sistemas dinámicos discretos

- Si en nuestro modelo hay algunas señales con un intervalo de muestreo diferente de otras, nos puede interesar conocer con detalle cuales
- Opción “Sample Time Display”



## Ejercicio 4.2

- Simule el siguiente sistema dinámico discreto:

$$x[n] = x[n - 1] + 25$$

Use el bloque “unit delay” y aplique el valor de la condición inicial donde corresponda



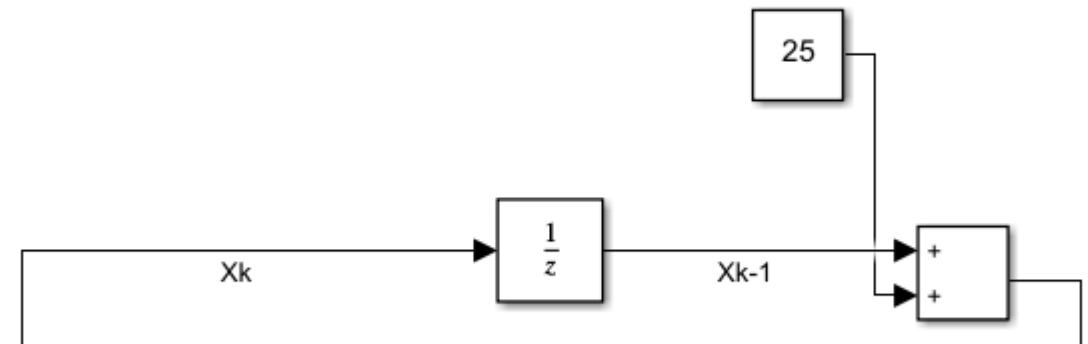
## Ejercicio 4.2



- Simule el siguiente sistema dinámico discreto:

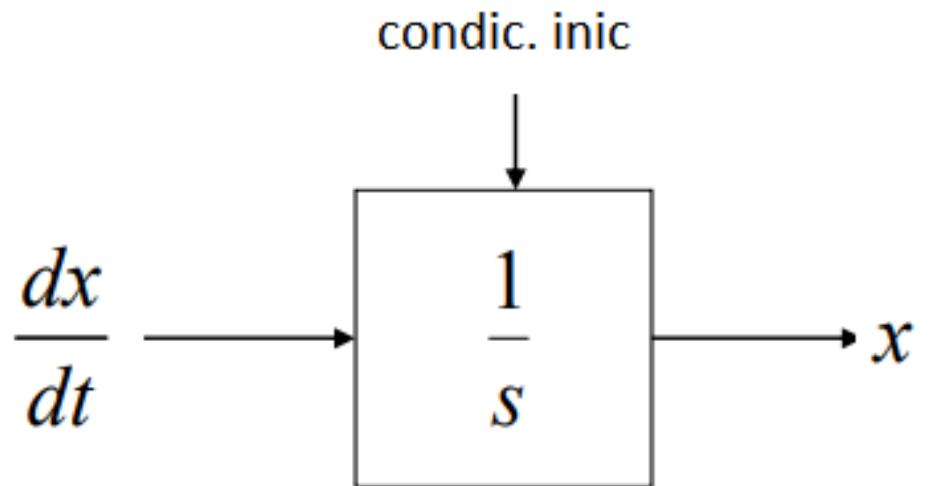
$$x[n] = x[n - 1] + 25$$

Use el bloque “unit delay” y aplique el valor de la condición inicial donde corresponda



# 5. Sistemas dinámicos continuos

- El modelado se hace en tiempo continuo
- Analogía con tiempo discreto:  
bloque integrator
  - Posibilidad de indicar condiciones iniciales
- Necesitaremos tantos bloques integradores como sea el orden del sistema dinámico



# Ejercicio 5.1

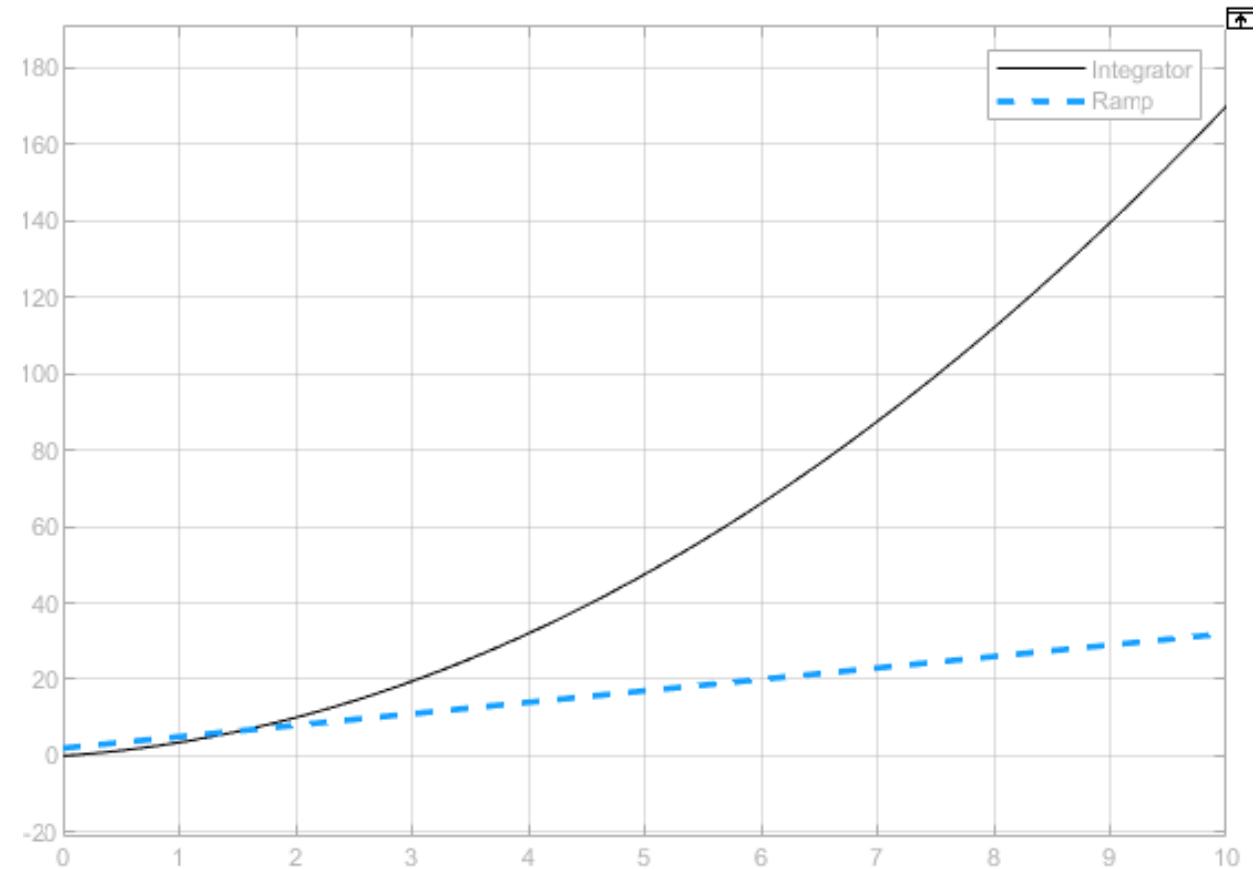
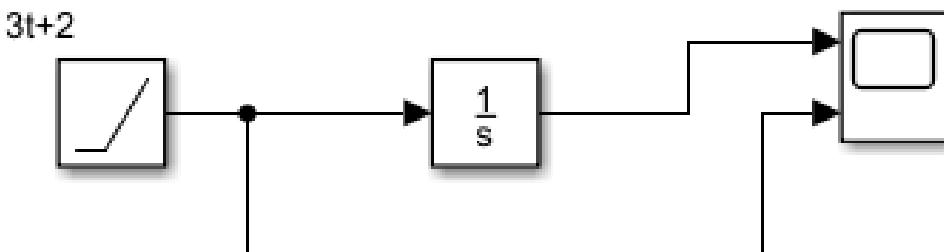
- Diseñe un modelo que calcule la integral de  $y(t) = 3t+2$
- Represente ambas señales.
- NOTA: puede usar el bloque ‘integrator’ y el bloque “ramp”



# Ejercicio 5.1



- Diseñe un modelo que calcule la integral de  $y(t) = 3t+2$
- Represente ambas señales.
- NOTA: puede usar el bloque ‘Integrator’ y el bloque “ramp”



## Ejercicio 5.2

- Modele el siguiente sistema dinámico en Simulink y represente las señales  $x(t)$  y su primera derivada

$$\frac{dx}{dt} = 5 \cos(3t)$$
$$x(0) = 1$$



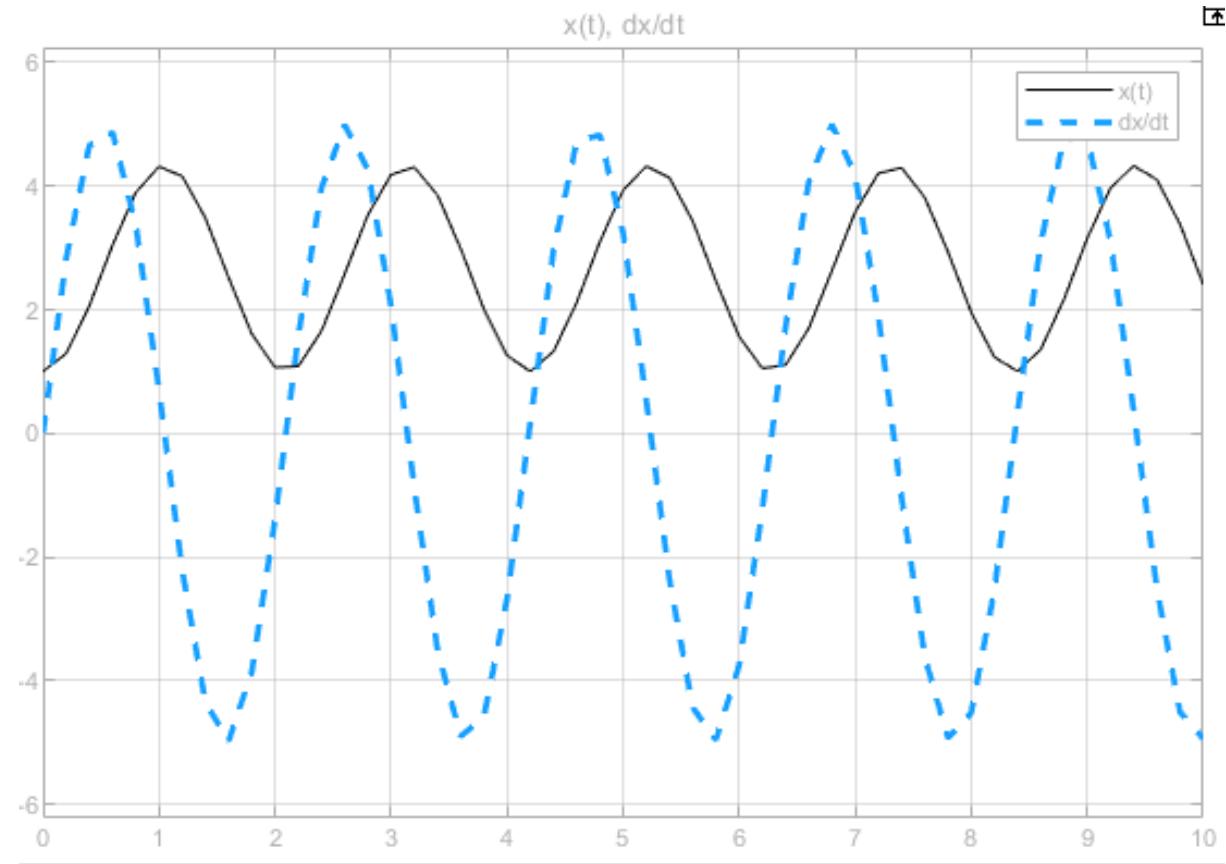
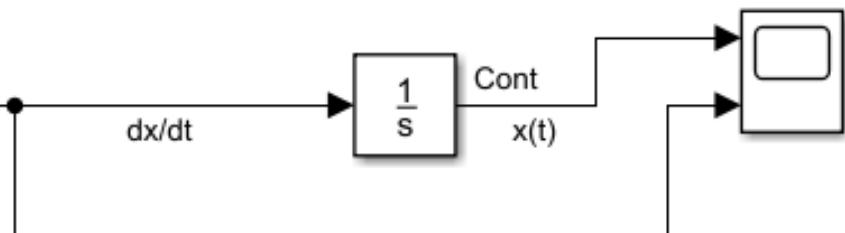
# Ejercicio 5.2



- Modele el siguiente sistema dinámico en Simulink y represente las señales  $x(t)$  y su primera derivada

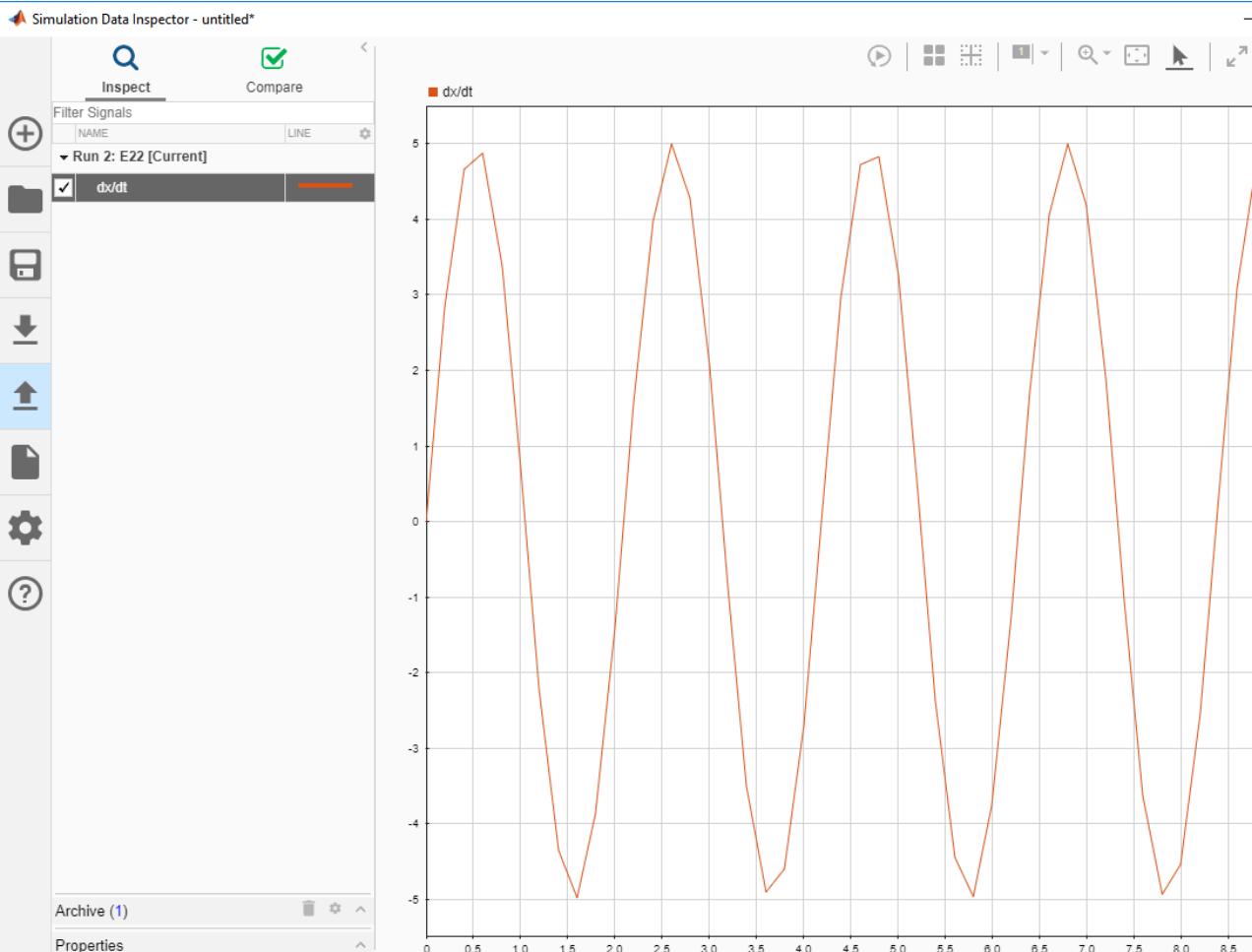
$$\frac{dx}{dt} = 5 \cos(3t)$$

$$x(0) = 1$$



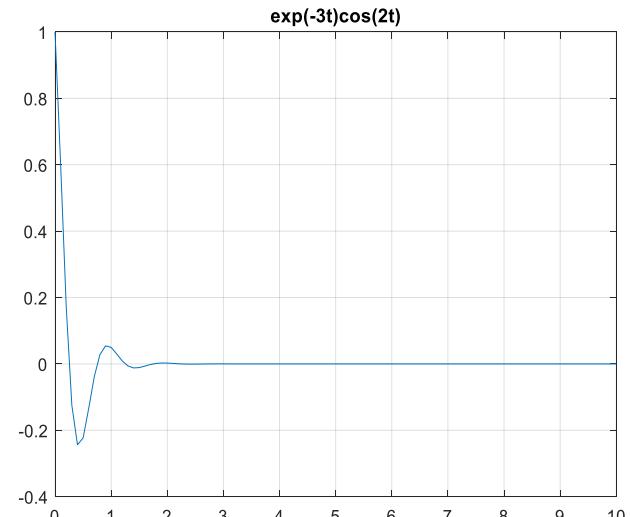
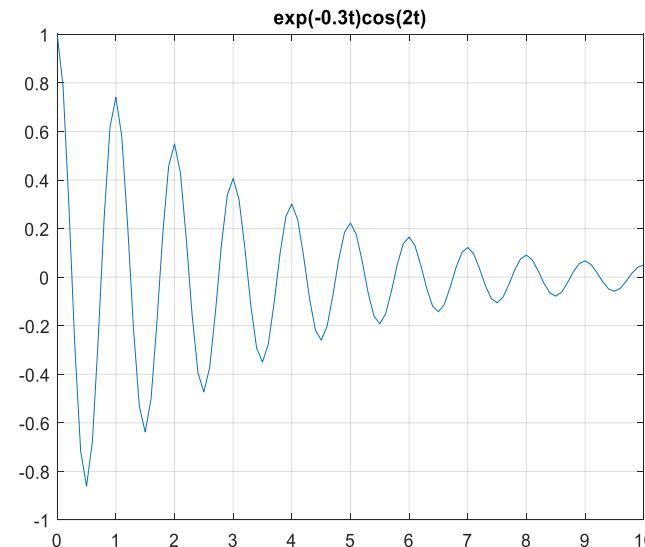
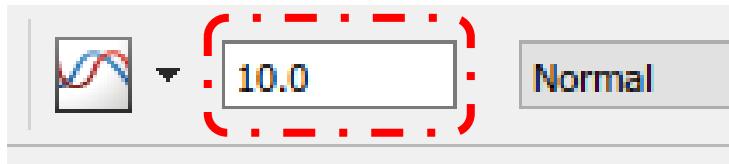
# 6. Inspección de señales

- La inspección de variables se puede realizar de diversas formas
  - “artesanales”, usando scope y otros
  - Exportando a MATLAB determinadas variables
  - “Simulation Data Inspector”
    - “Log selected signal”
  - ...



# 7. Tiempo de simulación

- Cuando cargamos un modelo, Simulink realiza una resolución numérica eligiendo un “solver”
- Simulink desconoce el tiempo más apropiado para la propia simulación → hay que indicárselo



## 8. Step size

- El paso de la simulación se puede especificar
- Por defecto se establece en 50 pasos por simulación

$$T_{step} = \frac{t_{stop} - t_{start}}{50}$$

- Es importante elegir bien este valor para optimizar el rendimiento de nuestras simulaciones
  - Es posible definirlo de forma diferente según bloques
    - Con ciertas limitaciones

# 9. Hardware en Simulink

- Simulink es una herramienta muy potente para control de HW
- Existe una gran variedad de dispositivos para los que existen drivers y soporte por parte de MATLAB



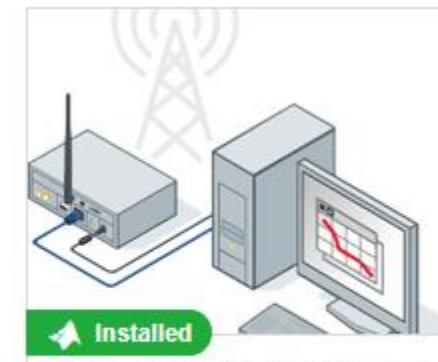
MATLAB Support Package  
for USB Webcams



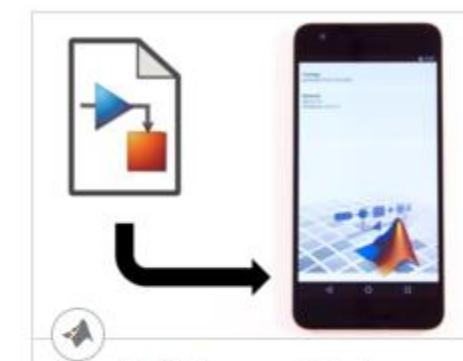
Rohde & Schwarz SMR  
Signal Generators



Rohde & Schwarz ESPI  
Test Receiver/Spectrum  
Analyzer



Communications Toolbox  
Support Package for USRP  
Radio



Simulink Support Package  
for Android Devices

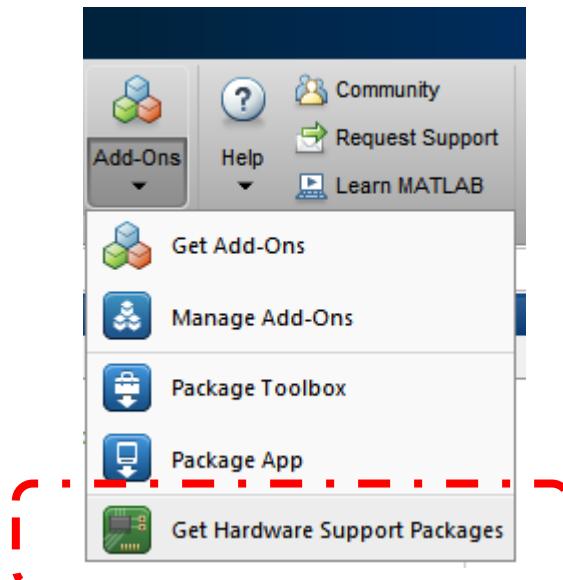
- Y un largo etcétera (ver '*Get Hardware Support Package*')

# 9. Hardware en Simulink

- La filosofía de trabajo es la misma con los dos dispositivos que abordaremos:
  - Raspberry PI → ‘mini PC’ desarrollado por la Raspberry Foundation.
  - Arduino → microcontrolador libre (HW & SW).
- Se requiere la instalación de drivers específicos en cada caso (Support Packages de MATLAB). →
  - problemas –no insalvables- con Windows 10.
  - Luego es necesario configurar esas instalaciones, conectando placas a nuestro PC
- Forma de trabajar: crear un modelo en Simulink.

# 9. Instalación y configuración de Arduino en MATLAB & Simulink

- Instalación

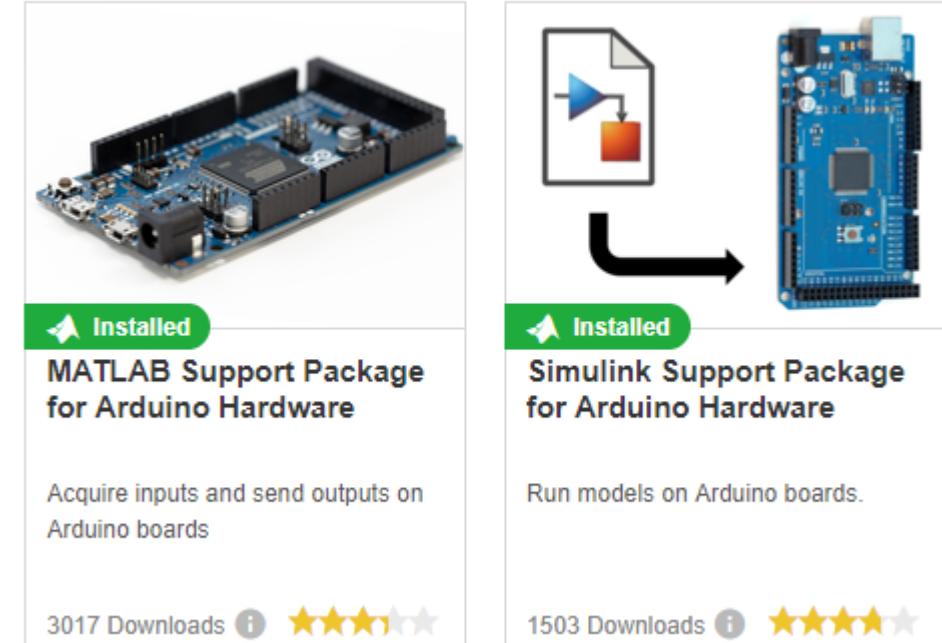


Si se instala el de Arduino, se genera una dependencia con el de MATLAB y se instalarán ambos

Pasos:

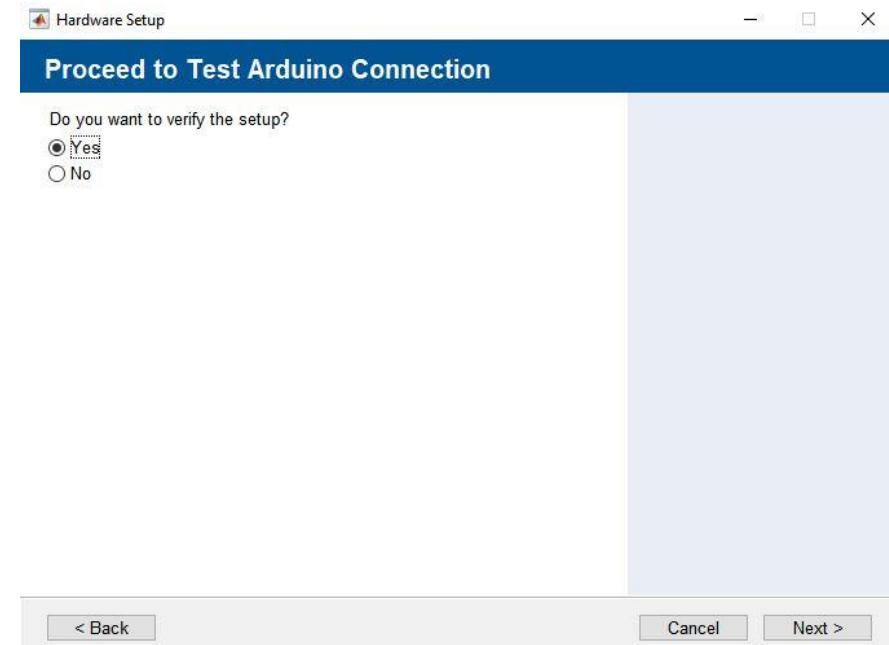
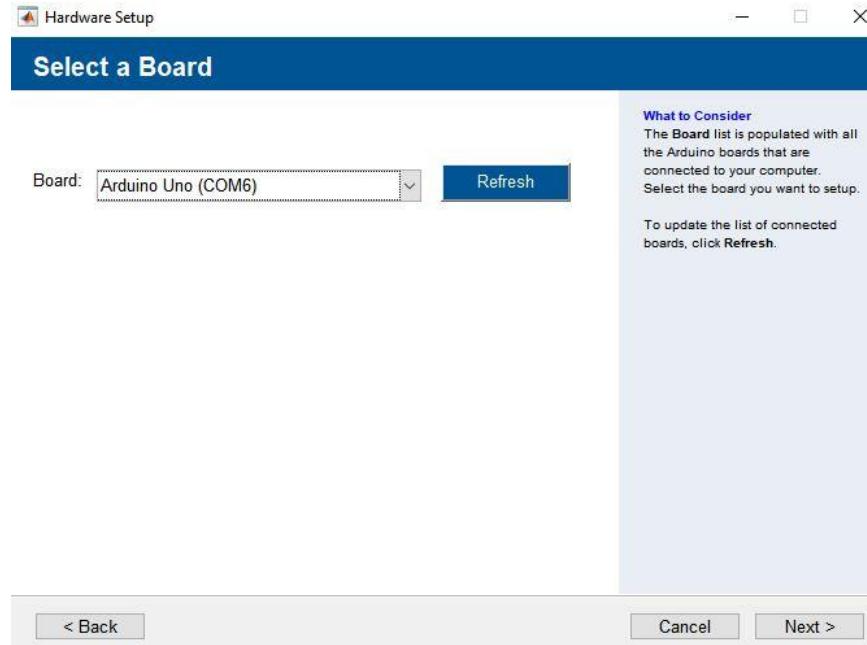
- Descarga
- Instalación
- Configuración (se requiere el HW)

## Hardware Support Packages (304)



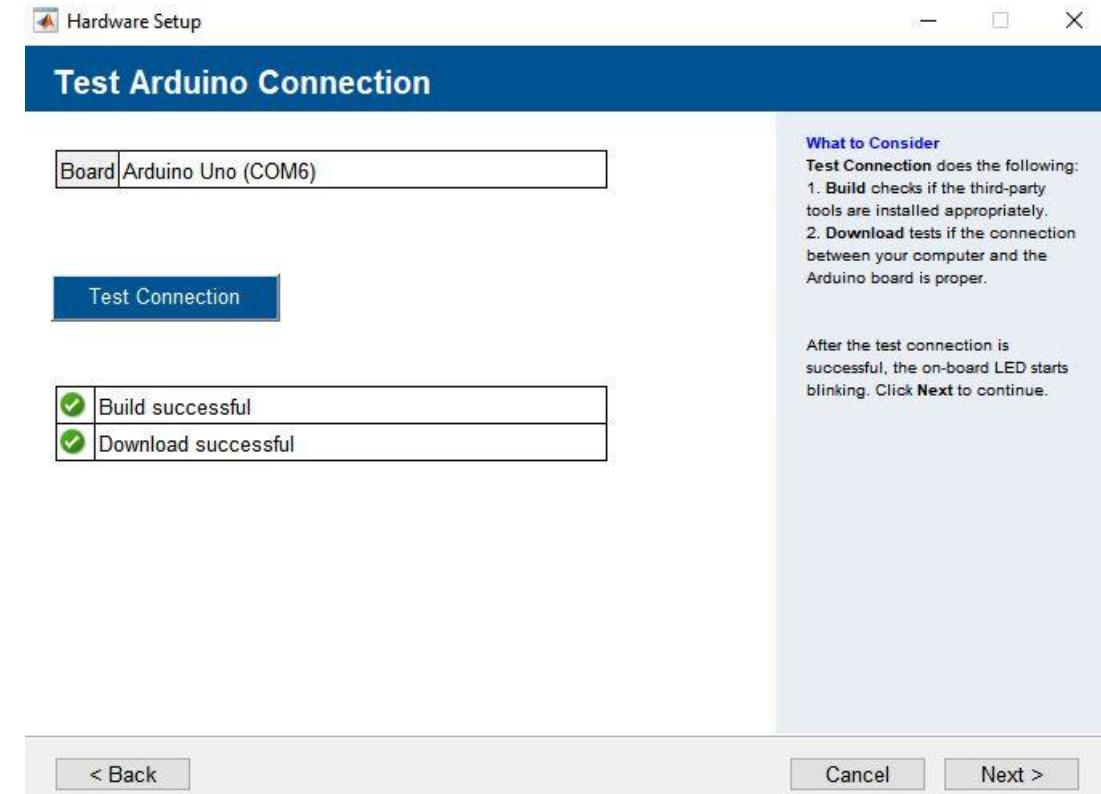
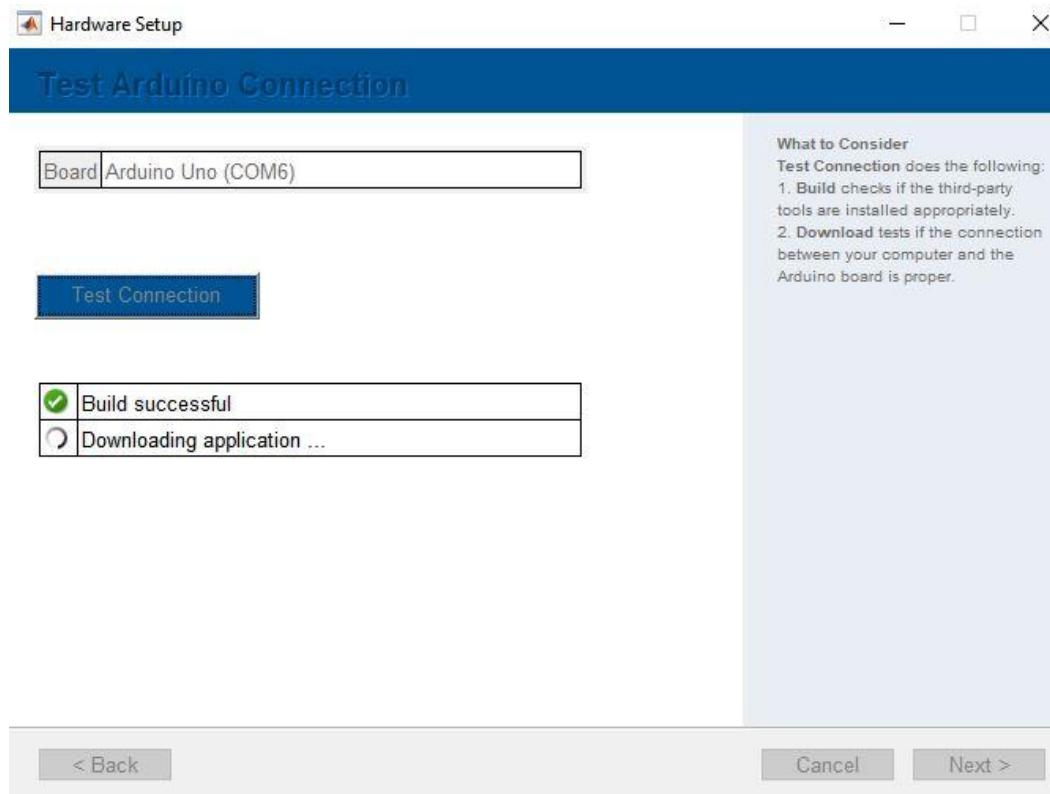
# 9. Instalación y configuración de Arduino en MATLAB & Simulink

- Configuración tras la instalación



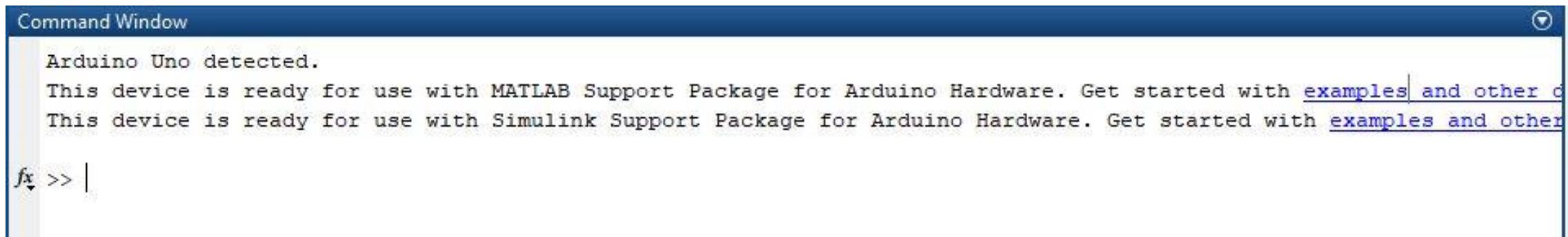
# 9. Instalación y configuración de Arduino en MATLAB & Simulink

- Configuración tras la instalación



# 9. Instalación y configuración de Arduino en MATLAB & Simulink

- Comprobación. Si se muestra este mensaje al conectar Arduino con MATLAB arrancado, la configuración es correcta



The screenshot shows the MATLAB Command Window with the title 'Command Window'. It displays the following text:  
Arduino Uno detected.  
This device is ready for use with MATLAB Support Package for Arduino Hardware. Get started with [examples](#) and other documentation.  
This device is ready for use with Simulink Support Package for Arduino Hardware. Get started with [examples](#) and other documentation.

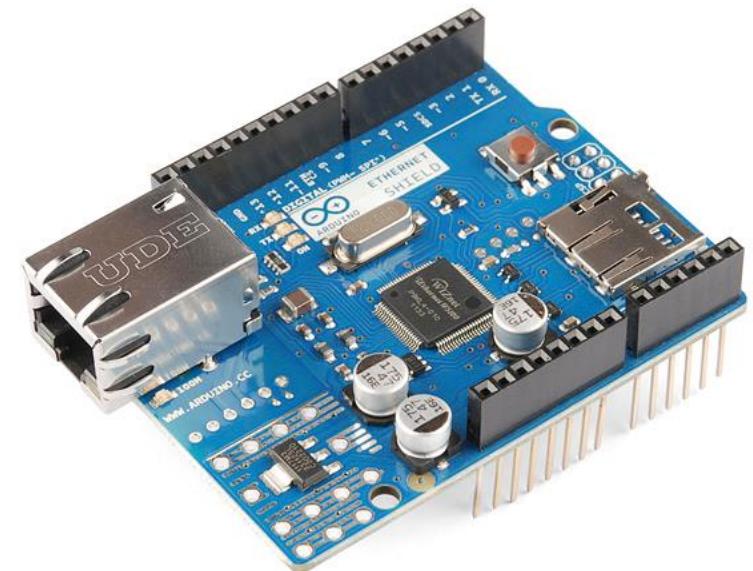
- Potencial problema con puerto COM. Resolución:
  - En Simulink: Simulation → Configuration parameters → Hardware implementation
    - Board → comprobar que seleccionamos nuestra placa Arduino
    - Target HW resources → seleccionar manualmente el puerto que corresponda

# 10. Resumen

- Filosofía de bloques
  - Admite encapsular → subsistemas
  - Bloque MATLAB function → definición de bloques sobre código MATLAB (c&p editor)
- Multitud de bloques predefinidos: para no “reinventar la rueda”
- Representación gráfica relativamente potente
- Memoria (WS) MATLAB → Simulink
  - Importacion/Exportación a WS MATLAB
- Salvado/importación de disco
- Facilidad para sistemas dinámicos
- Atención con tiempo de simulación y *sample time*

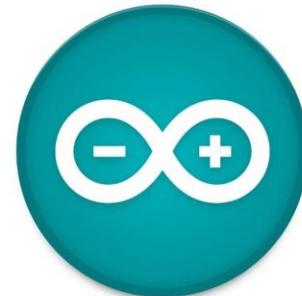
# Módulo 3: Programación de Arduinos con Simulink.

1. Placas Arduino. Descripción. Funcionamiento
2. Proyecto de Arduino usando Simulink.



# 1. Arduino

- Un microcontrolador es un circuito integrado programable, capaz de ejecutar las órdenes grabadas en su memoria.
- Compuesto de 3 bloques funcionales principales:
  - unidad central de procesamiento (CPU),
  - memoria (EEPROM)
  - periféricos de entrada y salida (I/O)
- Programación: ensamblador (o C).  
Posibilidad de compilación cruzada.



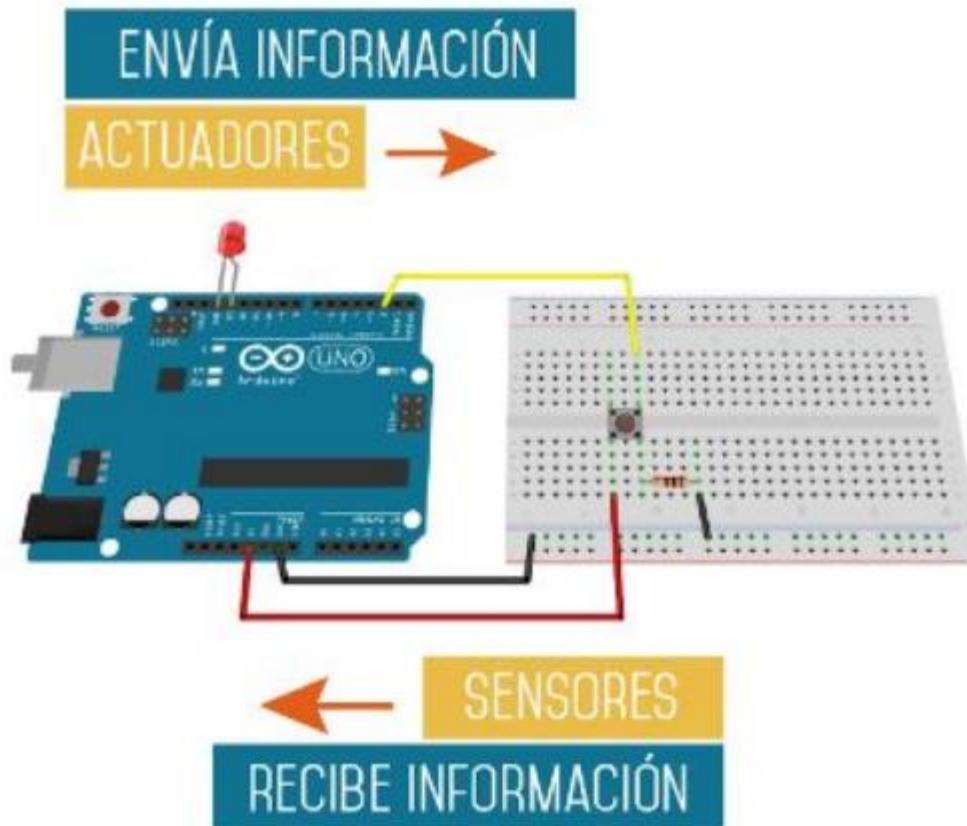
# 1. Arduino

- Familia de placas de desarrollo hardware para sensorización y control.
- Basados en micrонтroladores (y microprocesadores en algún caso).
- Admite placas de expansión (shields) para incrementar las funcionalidades de la placa Arduino: GSM, Ethernet, ...
- Programables desde PC (serie, USB,...). Uso autónomo.
- Bootloader que carga al inicio.
- Gran diversidad de modelos: Arduino Uno, Leonardo, Due, Yun, ...
- Dispositivos de gran aplicabilidad práctica.

# 1. Arduino

- Arduino es una plataforma de electrónica abierta, orientada a la creación de prototipos, basada en HW y SW libre
- Tipicamente:
  - Conexión de diversos sensores y actuadores a entradas/salidas digitales y analógicas
  - Programación: IDE Arduino
    - init() → inicialización de parámetros
    - loop() → bucle infinito
  - Hay más opciones para programación (Visual ino, Simulink, ...)
- Página oficial: <http://arduino.cc>

# 1. Arduino



```
void setup()
{
    Serial.begin(9600);
    Wire.begin();
    mpu.initialize();
    Serial.println(mpu.testConnection() ? F("IMU iniciado correctamente") : F("Error en la conexión del IMU"));
}

void loop()
{
    // Leer las aceleraciones y velocidades angulares
    mpu.getAcceleration(&ax, &ay, &az);
    mpu.getRotation(&gx, &gy, &gz);

    printRAW();

    delay(100);
}
```

# 1. Arduino

- El IDE de Arduino es el framework más utilizado y más potente
- El principal problema que tiene es la curva de aprendizaje → código
- Carga del SW, depuración, etc., son relativamente asequibles
- Ayuda en la web:
  - Gran disponibilidad de ejemplos
  - Comunidad de usuarios muy activa



# 1. Arduino

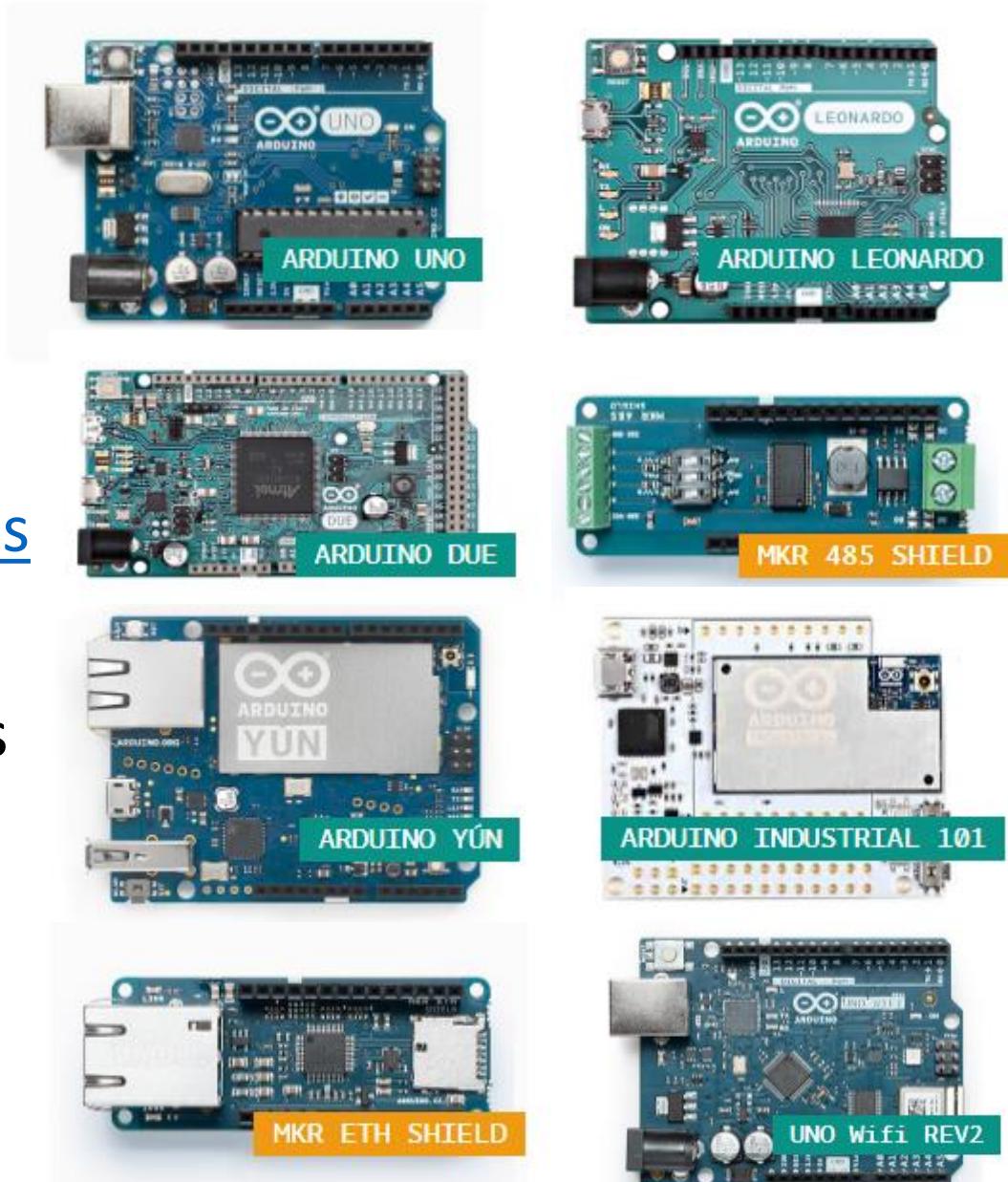
- Simulink como alternativa al IDE de Arduino
- Curva de aprendizaje mucho más suave
- Sinergias con MATLAB
- Configuración e instalación ya explicada

Hardware Support Packages (304)



# 1. Arduino

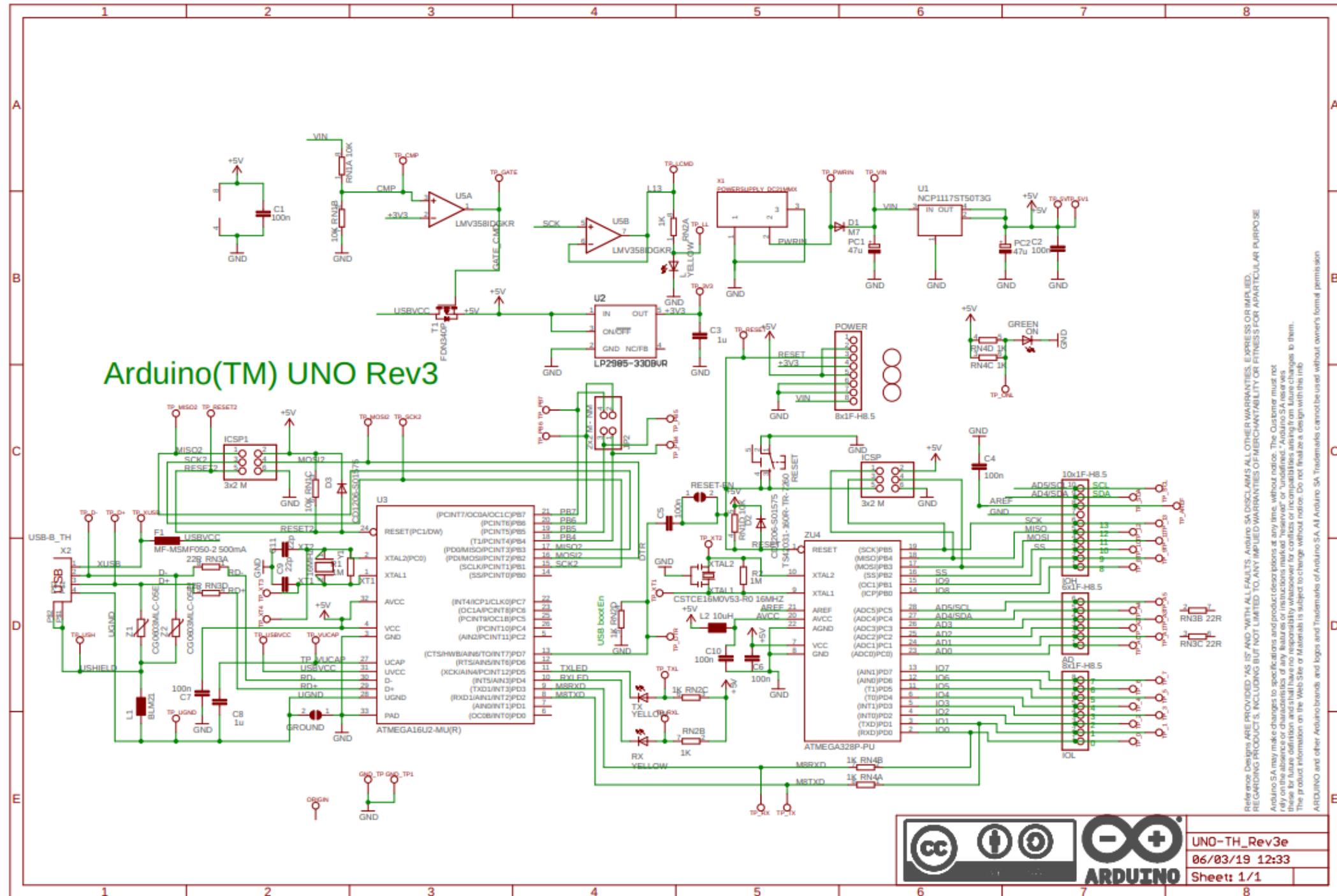
- Diversos productos dentro de la familia Arduino
- <https://www.arduino.cc/en/Main/Products>
- Nivel de entrada: Uno, Leonardo, ...
- Optimizados: Due, varios tipos de shields (WiFi, Ethernet, ...)
- IoT: Industrial 101, Ethernet, Yun, WiFi, SigFox, varios shields
- (Shields para expansión capacidades)

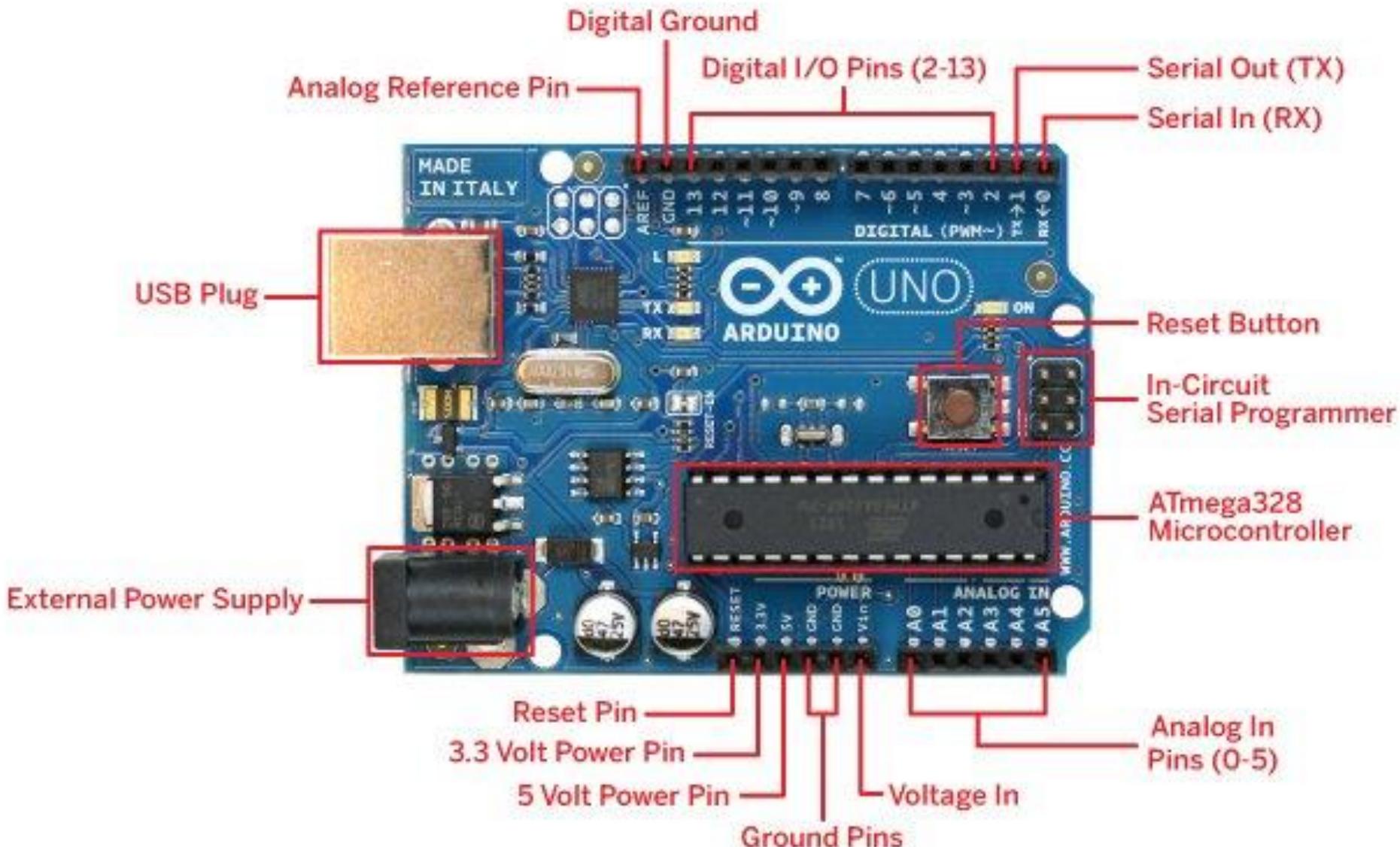


# 1. Arduino Uno

Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
LED_BUILTIN	13

- Modelo más popular en entorno educativo
- Lo usaremos en este laboratorio
- Existen alternativas no Arduino (i.e. Elegoo Uno) ☺
- 14 pines digitales (6 PWM)
- 6 pines analógicos
- @16 MHz
- EEPROM de 1K
- Protección sobrecorrientes (>500 mA) en puerto USB





# 1. Arduino

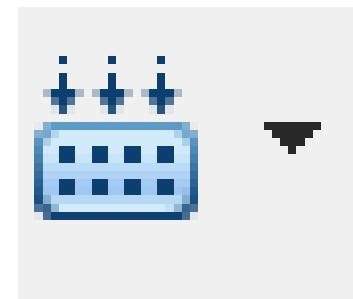
- Los arduino NO son sistemas de tiempo real
  - (\*) Hay algunas iniciativas para hacerlo, como ARTe
- Importante tener en cuenta los tiempos de cada tarea que llevamos a cabo.
- IDE Arduino tiene mejores métodos para monitorizar esto
- (Mucho prueba y error)

Time required to run various commands on an ATmega328-based Arduino running at 16 MHz

Various Arduino commands	Commands		Duration	
	μs	CPU cycles		
	Analog read	110.9	1775	
	Digital read (pin without PWM)	3.65	58	
	Digital read (pin with PWM)	4.53	72	
	Digital write (pin without PWM)	3.93	63	
	Digital write (pin with PWM)	4.84	77	
	Analog write	8.05	129	
	millis()	1.32	21	
	micros()	2.96	47	
	random()	140.3	2245	
Assembly commands	Direct read of 8 pins (e.g. PIND)	0.06	1	
	Direct read of 8 pins and store result to RAM (e.g. z = PIND)	0.19	3	
	Direct write to 8 pins (e.g. PORTD = B00001000)	0.06	1	
	Direct write to 8 pins from a byte stored in RAM (e.g. PORTD = z)	0.19	3	
	Skip a cycle (_asm_("nop\n\t"))	0.06	1	
Serial communication	Buffer free	Serial.write() @ 9600 baud	9.6	154
		Serial.write() @ 115200 baud	10.1	162
		Serial.print("Hello world") @ 9600 baud	113.5	1816
		Serial.print("Hello world") @ 115200 baud	119.0	1904
	Buffer full	Serial.write() @ 9600 baud	1039	16621
		Serial.write() @ 115200 baud	83.6	1338
		Serial.print("Hello world") @ 9600 baud	11439	183021
		Serial.print("Hello world") @ 115200 baud	933.8	14941

## 2. Arduino en Simulink

- Haremos nuestro modelo en Simulink.
  - Definición de pines
  - Lógica para actuadores / sensores
- Carga en dispositivo → “Deploy to hardware” (en vez de “Run”, como en las simulaciones)
- Fases de la carga:
  - Compiling...
  - Initializing...
  - Building...
  - Ready

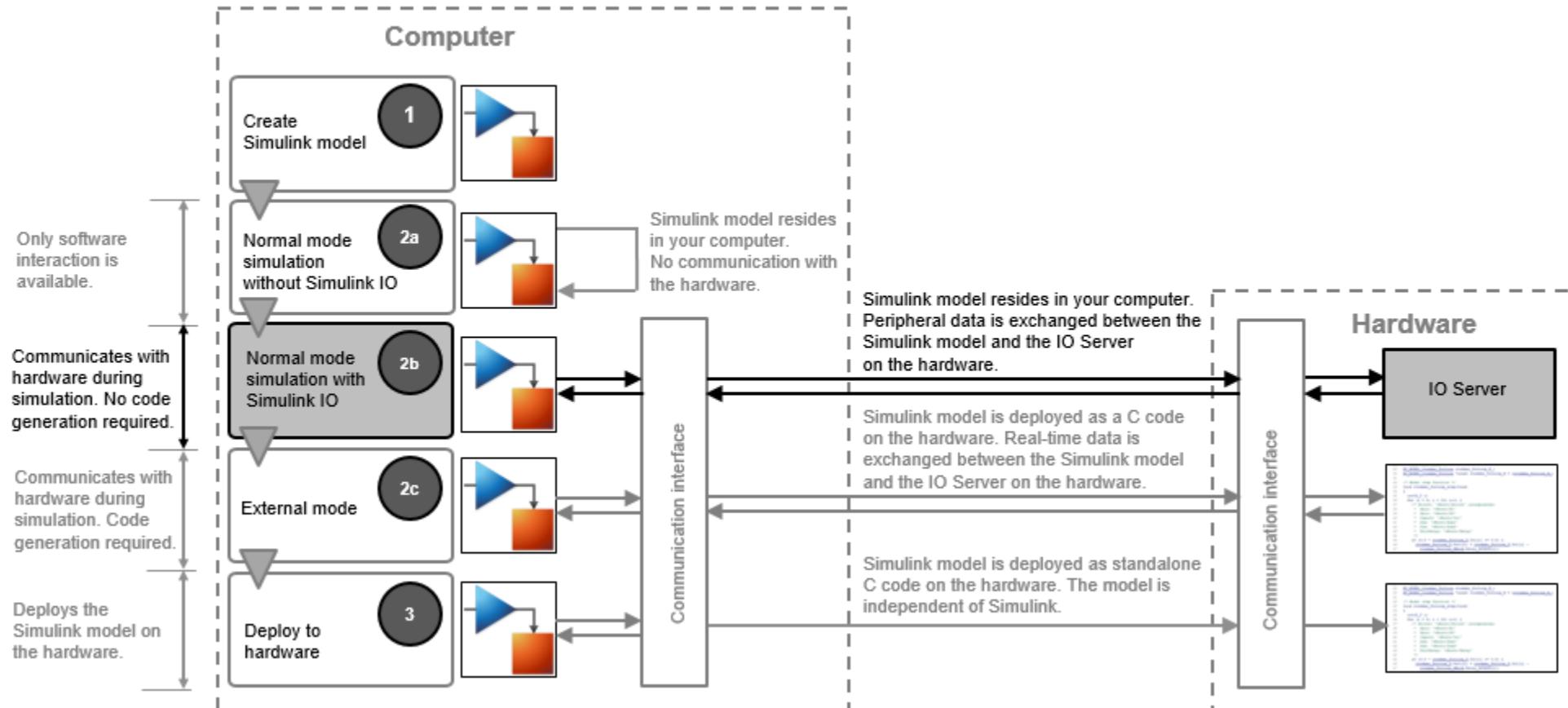


Buena política: simular primero (“Run”)

<https://es.mathworks.com/help/supportpkg/arduino/ug/simulink-io.html>

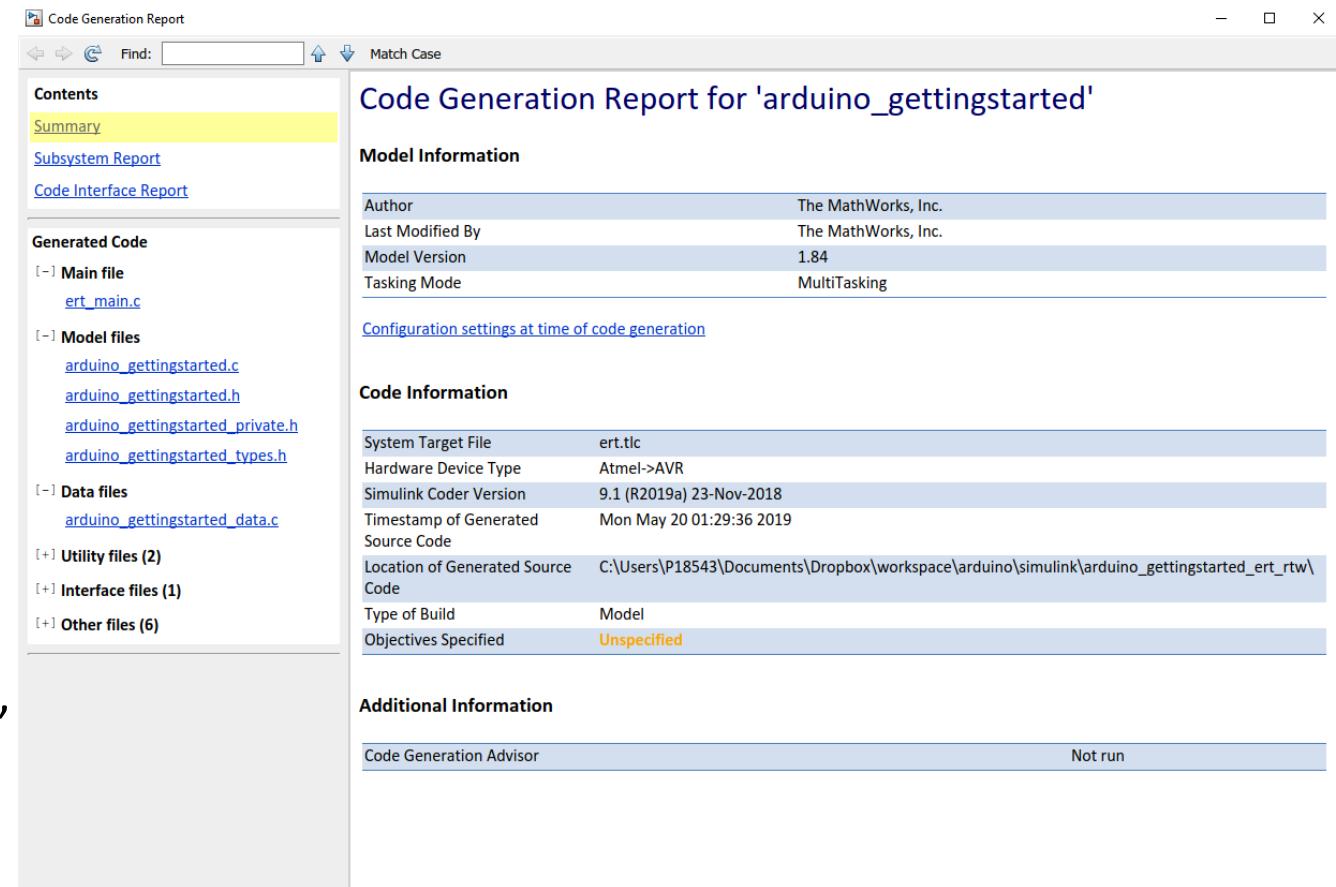
## 2. Arduino en Simulink (modelo HW en general)

Model-Based Design Workflow

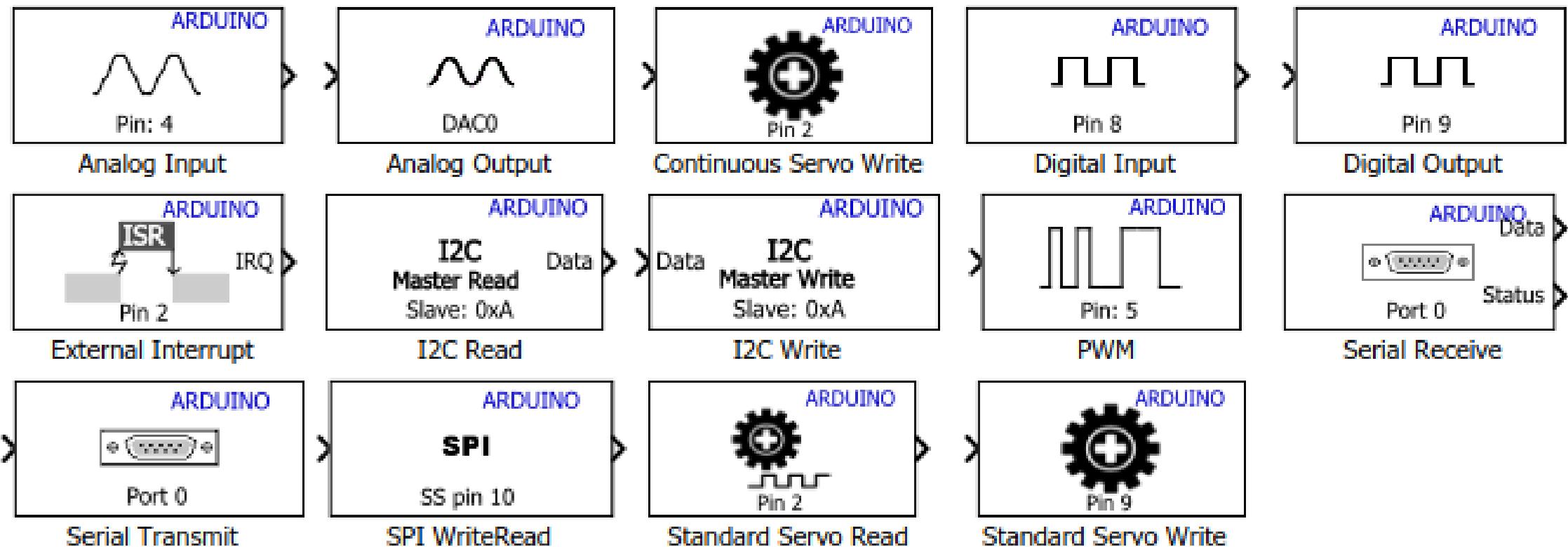


# 2. Arduino en Simulink

- En caso de éxito, se genera un informe del código cargado
  - Versión
  - Espacio ocupado
  - Archivos (.c, .h, etc.)
- Troubleshooting:
  - Carga de SW con dispositivo conectado
  - Puerto COM correcto
    - Ver “administrador de dispositivos”
  - Conflictos de rutas
  - ...



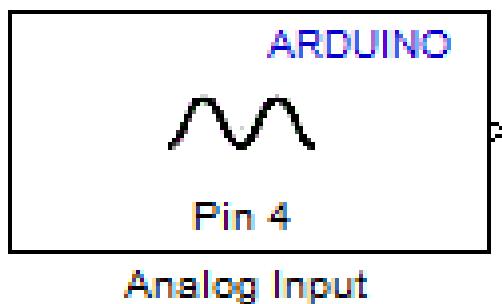
## 2. Arduino en Simulink. Common blocks



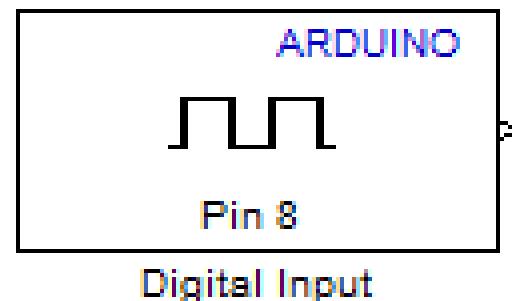
Ayuda: <https://es.mathworks.com/help/supportpkg/arduino/common.html>

## 2. Arduino en Simulink

### Analog input



### Digital input



- Mide la tensión en el pin que corresponda en relación al de referencia (por defecto es 5 V)
- Devuelve un valor entre 0-1023

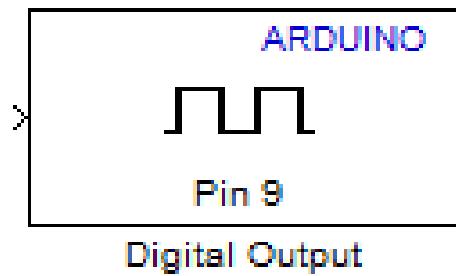
- Obtiene el valor LOGICO de un pin digital ( $0\text{ V} = \text{'0'}$ ;  $5\text{ V} = \text{'1'}$ ). Boolean

## 2. Arduino en Simulink



### Digital output

- Pone en el pin correspondiente el valor LOGICO (0 V – 5 V)



### PWM (Pulse Width Modulation)

- Pone en el pin correspondiente una señal cuadrada con un ciclo de trabajo (“ancho”) proporcional al valor de entrada (0-255)
- No todos los pines soportan PWM
- Hay pines pensados para PWM (se consiguen mejores frecuencias). En Arduino Uno son los pines 5 y 6
  - Ver asignación de pines

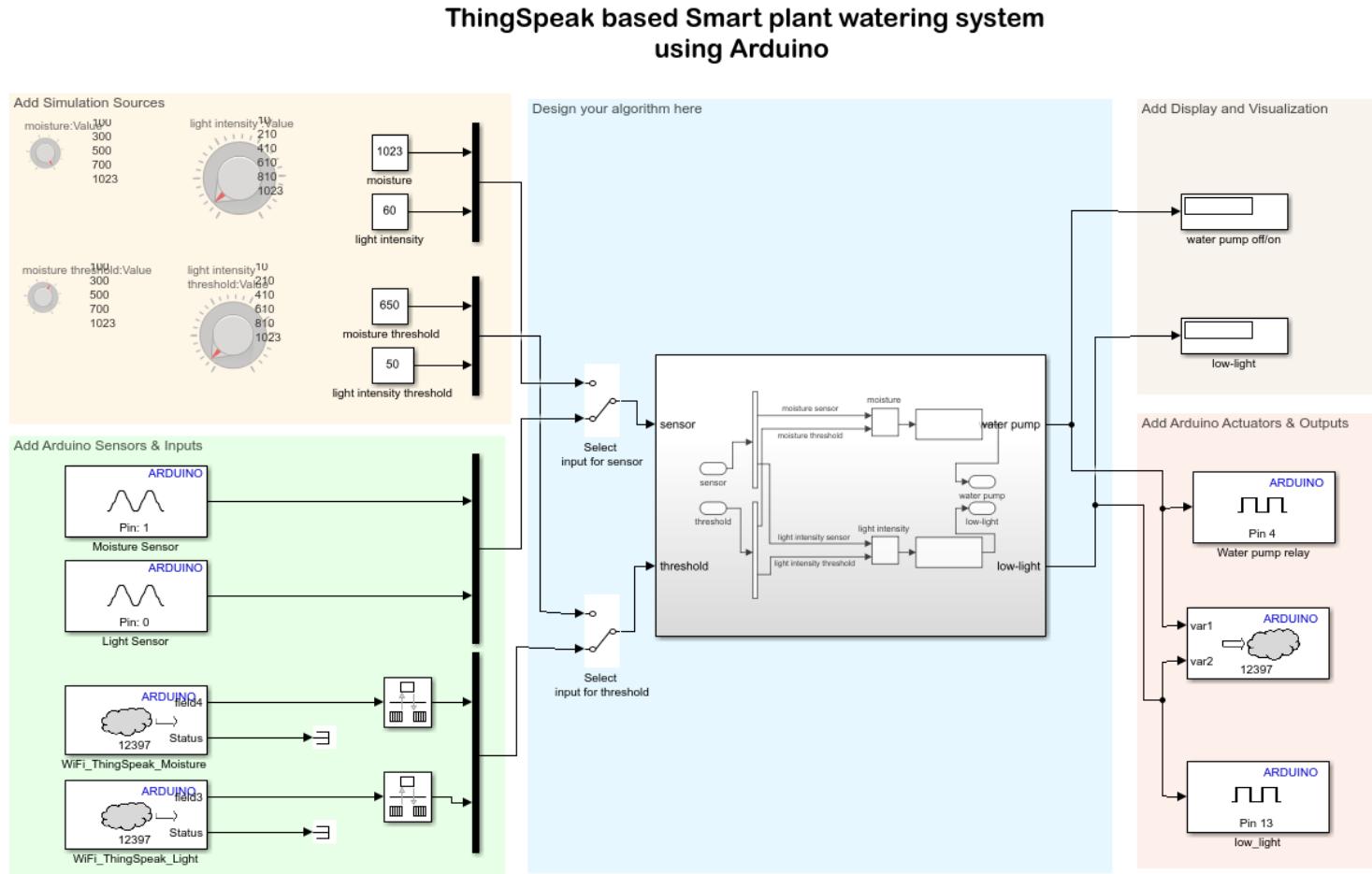
## 2. Arduino en Simulink

- En pura teoría, el mínimo ‘sample time’ es  $1\mu\text{s}$  (¿?)
- No asignar el mismo pin a diferentes bloques → conflicto
- 1 bloque común por pin
  - Verificar asignación correcta de pines en la placa Arduino que tengamos
- Lógica asociada a cada pin
- Simulación (depurar errores) → “*deploy to hardware*” (fuego real ☺)
- Problema con capacidad memoria Arduino. No todos los modelos caben

```
## Build procedure for model: 'arduino_LED_jmg1' aborted due to an error.  
The following error occurred during deployment to your hardware board:  
The generated code exceeds the available memory on the processor. It uses 19.4% of available program  
memory and 137.9% of available Data memory.  
Component: Simulink | Category: Block diagram error
```

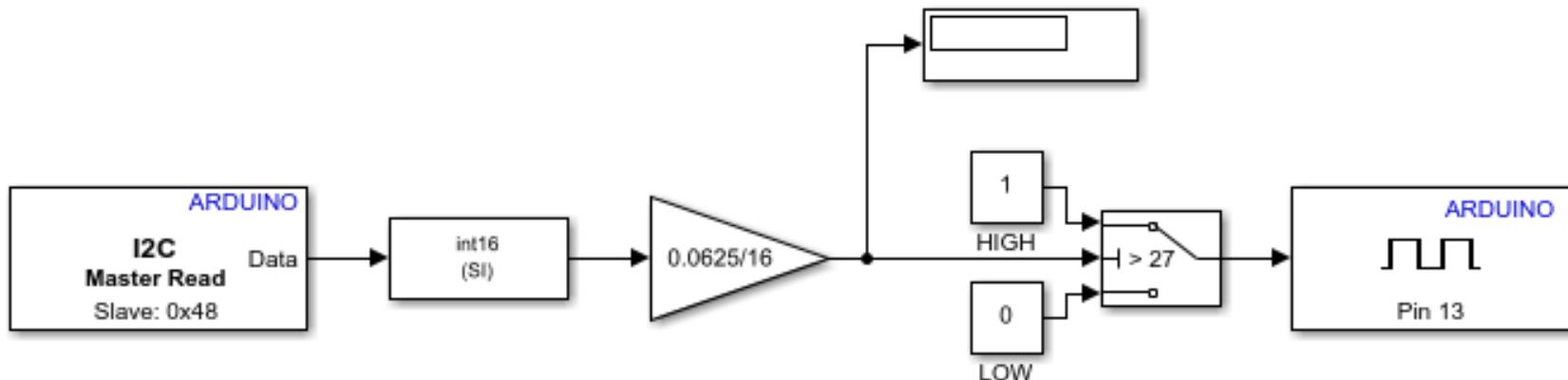
# 2. Arduino en Simulink. Filosofía de trabajo

Ejemplo para riego  
de plantas



## 2. Arduino en Simulink. Filosofía de trabajo

**Read temperature from an I<sup>2</sup>C based sensor  
using Arduino Hardware**

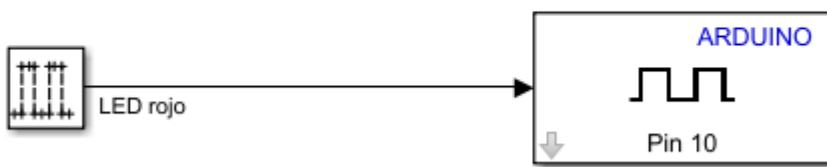


## 2. Ejemplos Arduino con Simulink

1. Iluminación de 1-2 LEDs con Arduino Uno
2. Iluminación intermitente de un LED
3. Cambio de ciclo en iluminación de un LED
4. Ajuste de Intensidad (PWM) de un LED
5. Leer de un sensor analógico o digital:
6. Trabajar con un pulsador
7. Semáforo cíclico (3 LEDs colores en secuencia cíclica)
8. Semáforo controlado por un pulsador → sensor & algoritmo & actuador
9. Bonus: acelerómetro

# 3. Iluminación de un LED

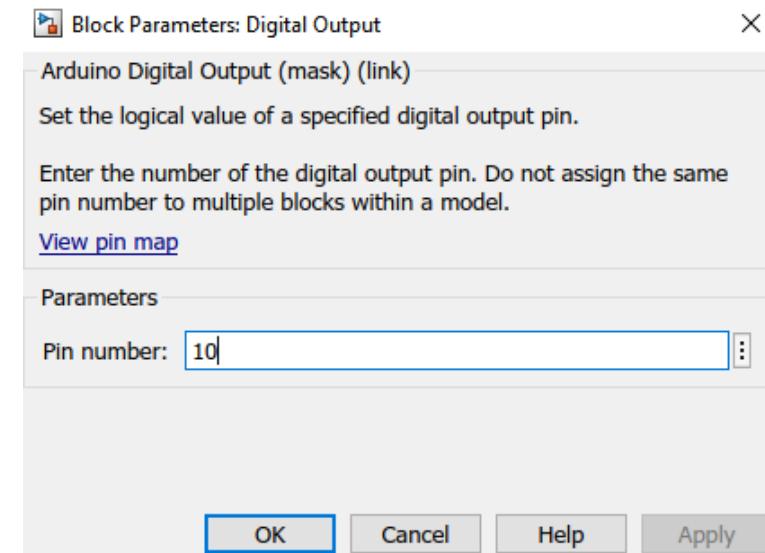
Ejemplo 1: 1 LED. Variar fases y ciclos.



- Necesidad de conectar una resistencia para limitar la corriente por el LED
  - $1\text{ K}\Omega$

- Consultar pines disponibles
- Bloque Digital output
  - Valor de salida es boolean
- Source: pulse generator
  - Control del “ciclo de trabajo”

# 3. Iluminación de un LED



Parameters

Pulse type: Sample based

Time (t): Use simulation time

Amplitude:

1

Period (number of samples):

12

Pulse width (number of samples):

12

Phase delay (number of samples):

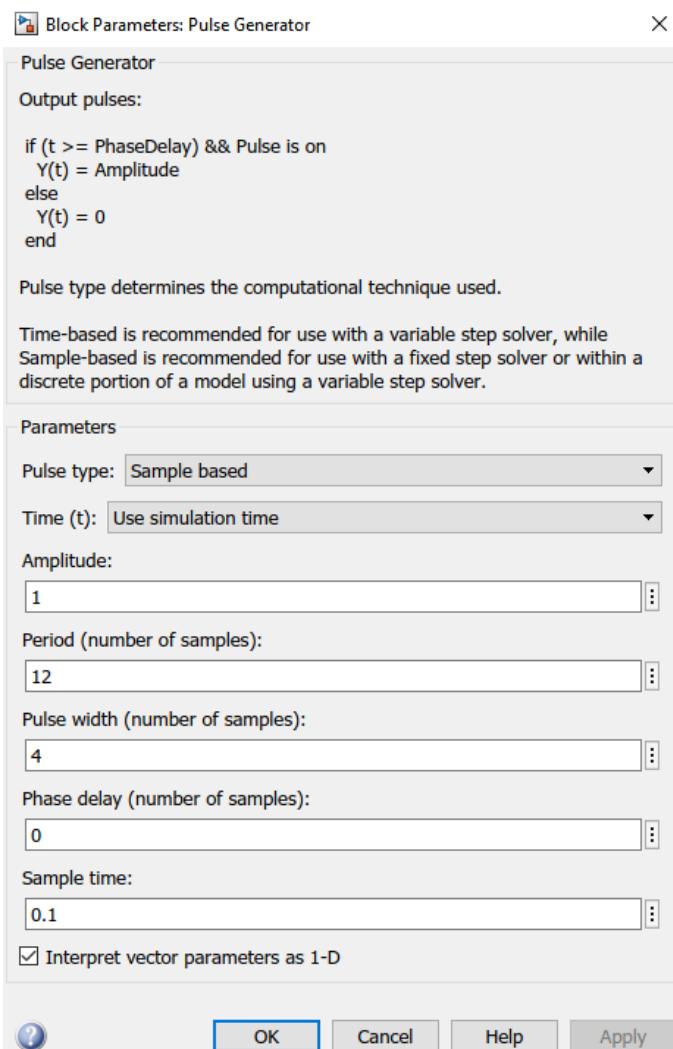
0

Sample time:

0.1

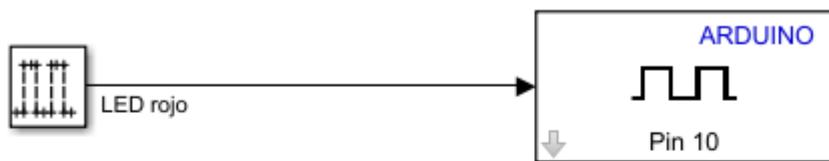
Blocks		
	Uno	Nano 3.0
Digital Input	0-13	0-13
Digital Output	0-13	0-13
Analog Input	0-5	0-7
Analog Output	N/A	N/A
PWM	3,5,6,9,10,11	3,5,6,9,10,11
Standard Servo Read	0-13	0-13
Standard Servo Write	0-13	0-13
Continuous Servo Write	0-13	0-13
External Interrupt	2,3	2,3
SPI Slave Select (SS)	0-10	N/A
SPI MOSI	11/ ICSP-4	N/A
SPI MISO	12/ ICSP-1	N/A
SPI SCK	13/ ICSP-3	N/A
I2C SDA	A4	D4
I2C SCL	A5	D5
Serial Receive	Port 0: pin 0 Port 1: N/A Port 2: N/A Port 3: N/A	Port 0: pin 0 Port 1: N/A Port 2: N/A Port 3: N/A
Serial Transmit	Port 0: pin 1 Port 1: N/A Port 2: N/A Port 3: N/A	Port 0: pin 1 Port 1: N/A Port 2: N/A Port 3: N/A
TCP/IP Receive	10,11,12,13	10,11,12,13
TCP/IP Send	10,11,12,13	10,11,12,13

# 4. Iluminación intermitente de un LED



- Parámetros:
  - Período
  - Ancho del pulso
  - Desfase
  - Tiempo de muestreo
- Probemos a bajar el Ts al mínimo posible

# 5. Varios LEDs

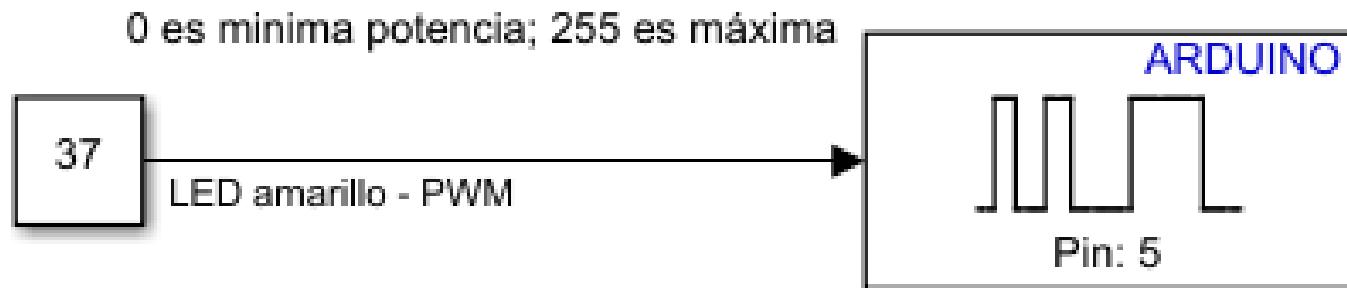


Ejemplo 2: LED. Añadir otro LED en pin 12



- Cada LED tiene sus propios parámetros
  - Gestión independiente
- Importante no poner distintos controladores en el mismo puerto (!)

# 6. Ajuste de Intensidad de un LED

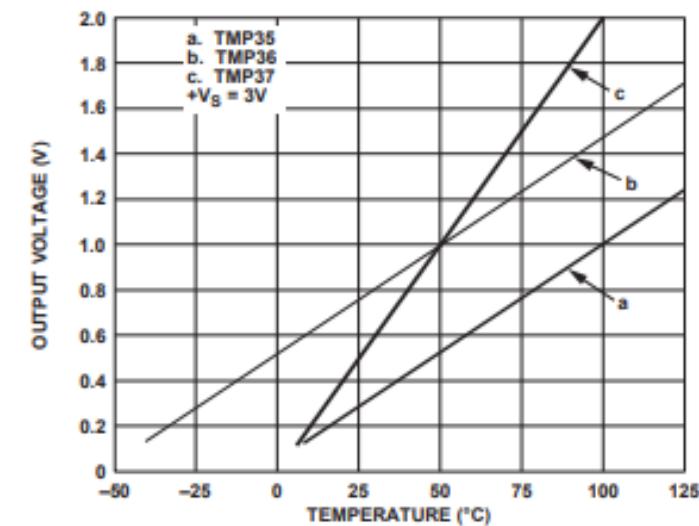
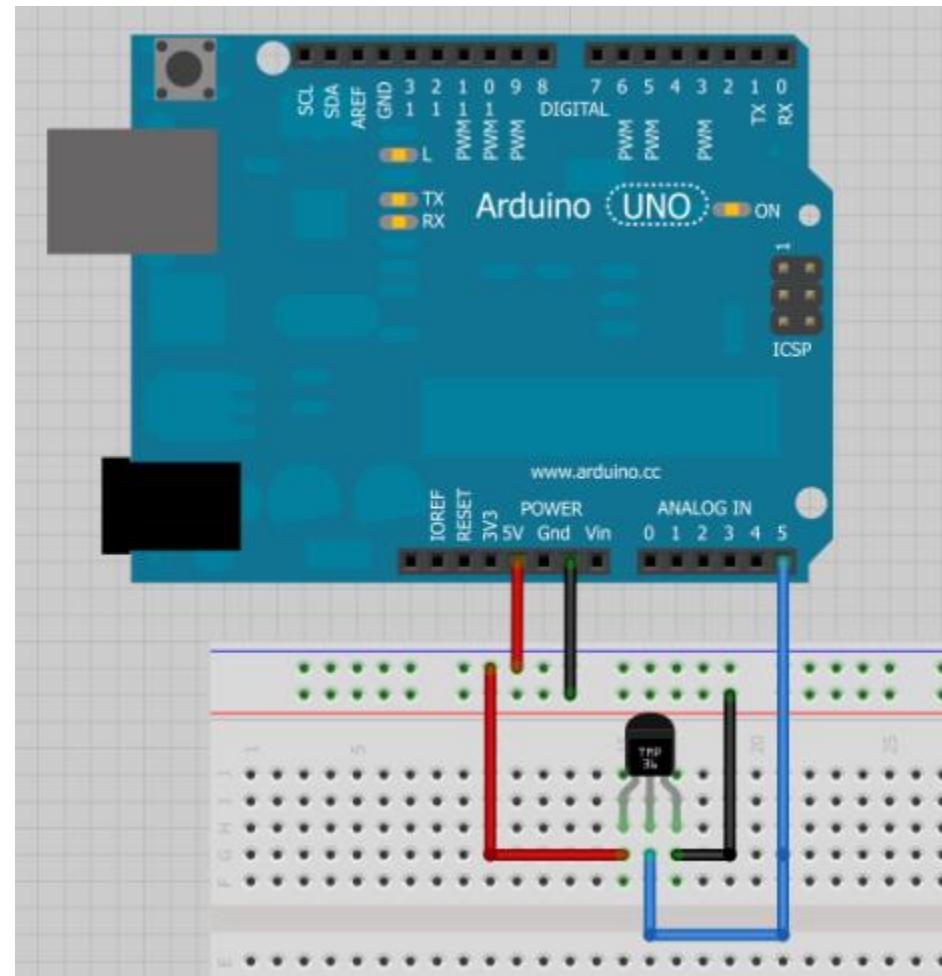


- Modulación por ancho de pulso
- Entrada entre 0-255, proporcional a la intensidad
- Importante: solo determinados puertos admiten PWM (ver hoja)

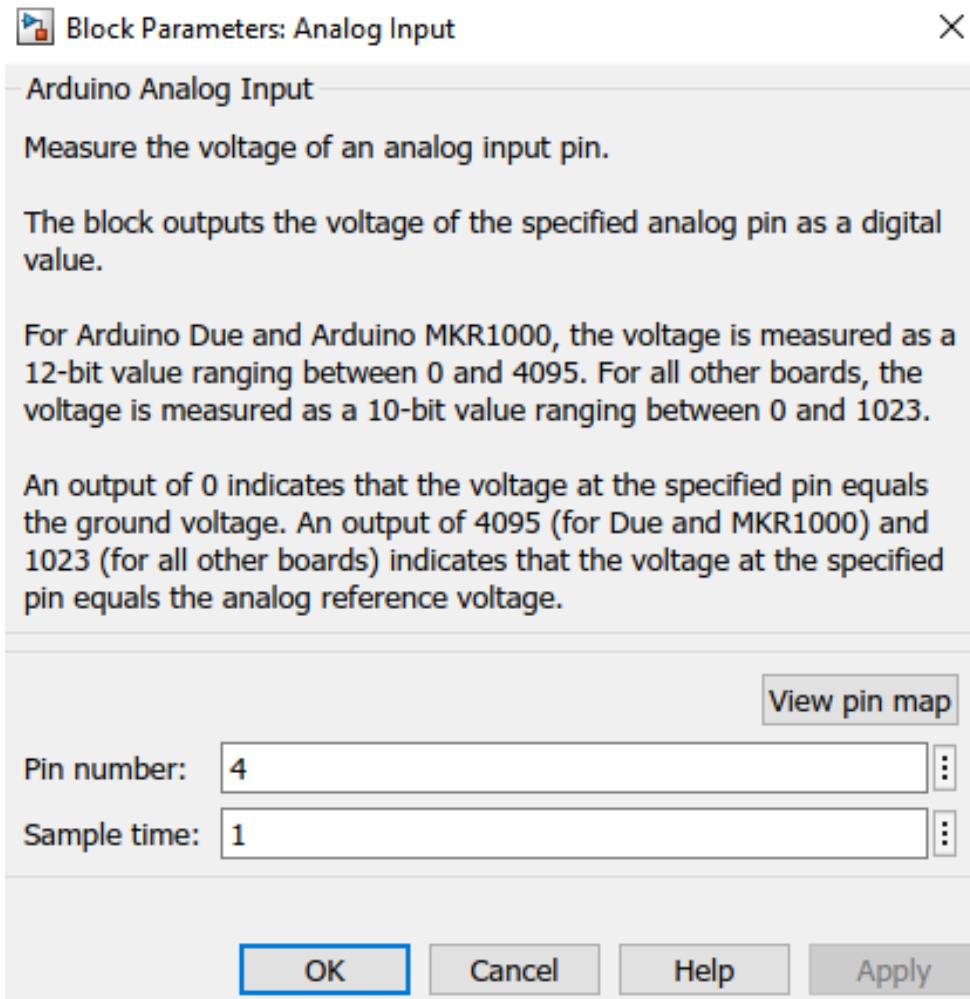
Blocks		Uno
Digital Input	0-13	
Digital Output	0-13	
Analog Input	0-5	
Analog Output	N/A	
User Selectable Pins	PWM	3,5,6,9,10,11
	Standard Servo Read	0-13

# 7. Leer de un sensor analógico (TMP36)

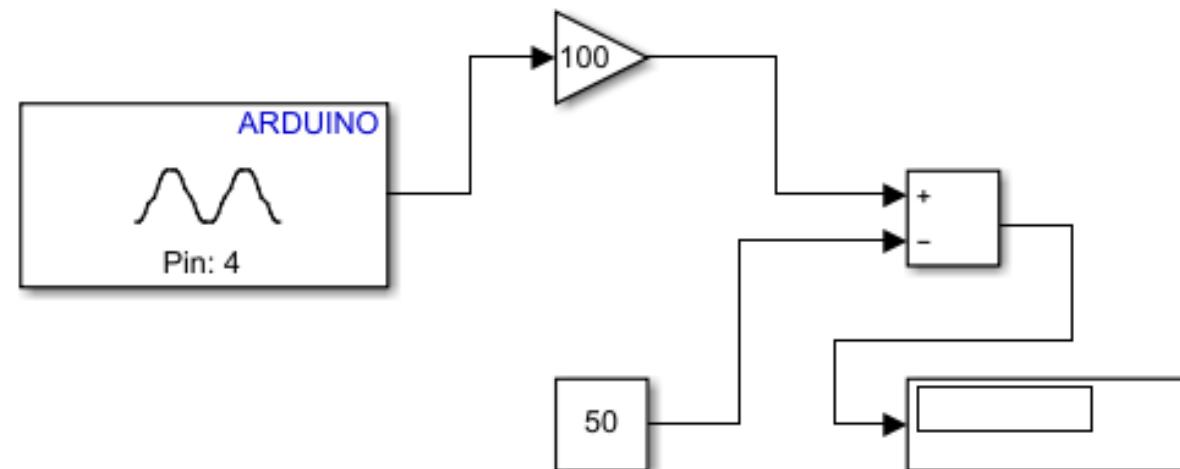
- Ejemplo analógico:
- Prestamos atención al pin que corresponda (son varios posibles)
- “procesado” valor leído para obtener medida
  - Ver hoja características sensor



# 7. Leer de un sensor analógico (TMP36)



TMP36: sensor 3 patas (GND, 5V, pin A4)  
Temp\_C = 100V-50



# 7. Leer de un sensor analógico

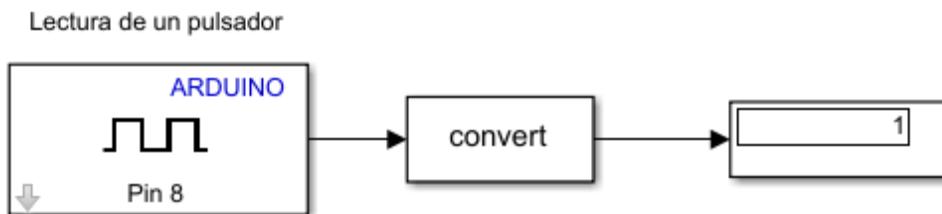
- Bonus: lectura con MATLAB

```
a = arduino  
volt = readVoltage(a, 'A7')
```

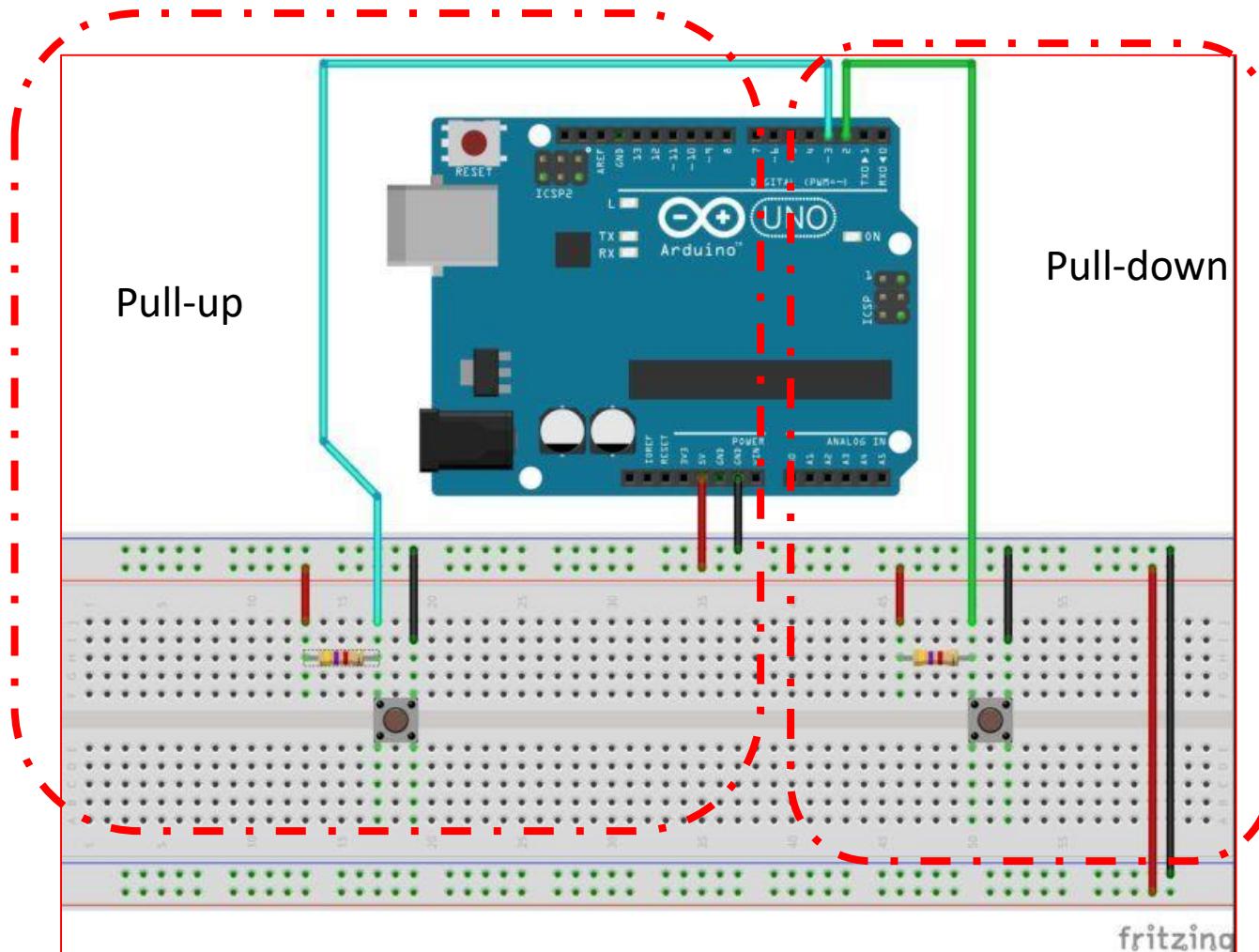
- Problemática potencial en gestión de la comunicación (o solo Simulink o solo MATLAB)
  - Se soluciona borrando el objeto arduino ('a') antes de usarlo en otro sitio

# 8. Trabajar con un pulsador

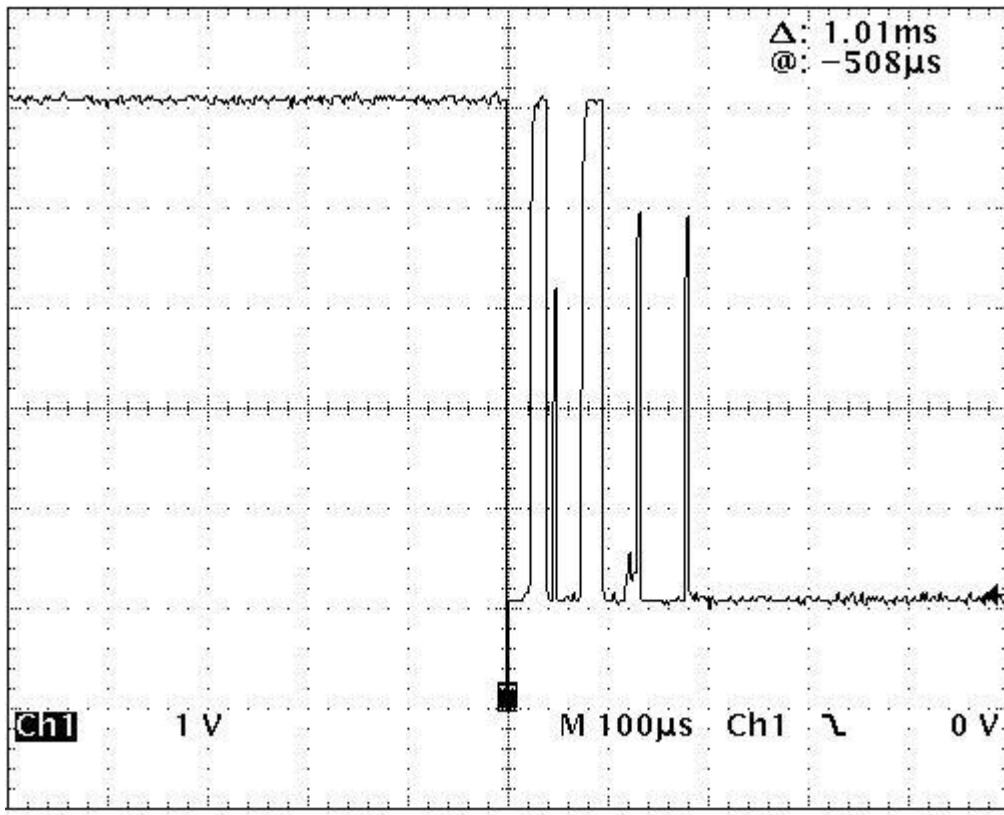
- Resistencia de pull-up (o down) necesaria



- Entrada digital
- Conversión para representación



# 8. Trabajar con un pulsador



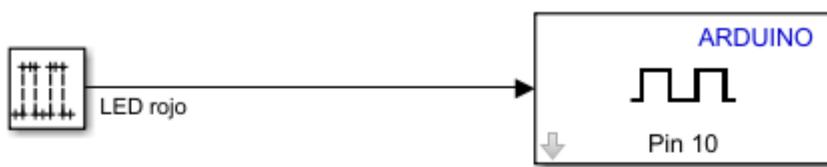
- Necesidad *futura* de eliminar ruido y rebotes
- Especialmente severo si gobernamos por interrupción
- Lo ideal es:
  - Detectar flanco subida/bajada
  - Inhibir un tiempo T antes de volver a leer

## 9. Semáforo cíclico

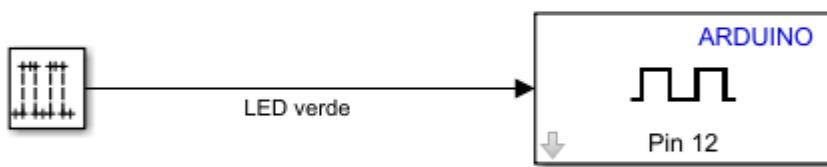
- Poner 3 LED, uno de cada color cada uno en una salida digital de Arduino
- Con “Pulse Generator”, asignar subperiodos dentro de un ciclo a cada LED, de forma que solo brille uno.
- Se puede ajustar la intensidad de cada uno.
- Sin gestión de “exclusividad”



# 9. Semáforo cíclico



Este LED tiene un desfase de 4 Ts con respecto al rojo



Este LED tiene un desfase de 8 Ts co nrespecto al rojo



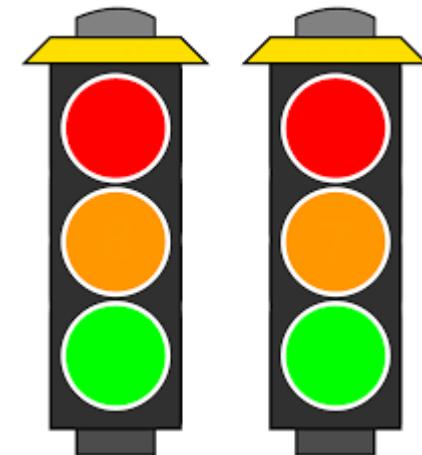
Parameters
Pulse type: Sample based
Time (t): Use simulation time
Amplitude:
1
Period (number of samples):
12
Pulse width (number of samples):
4
Phase delay (number of samples):
0
Sample time:
0.1

Parameters
Pulse type: Sample based
Time (t): Use simulation time
Amplitude:
1
Period (number of samples):
12
Pulse width (number of samples):
4
Phase delay (number of samples):
8
Sample time:
0.1

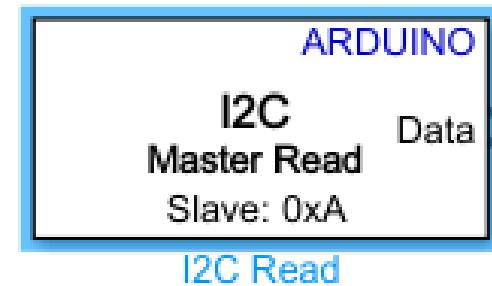
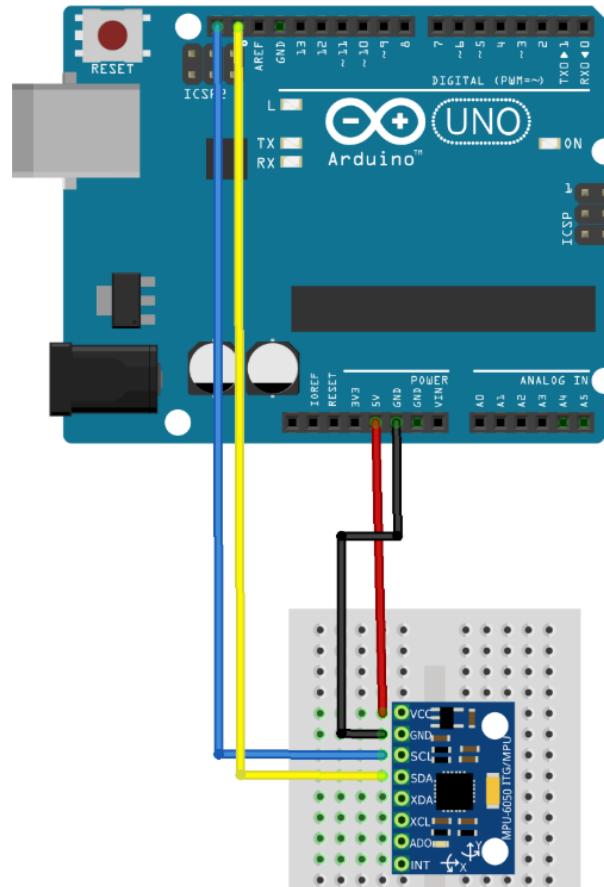
Parameters
Pulse type: Sample based
Time (t): Use simulation time
Amplitude:
1
Period (number of samples):
12
Pulse width (number of samples):
4
Phase delay (number of samples):
4
Sample time:
0.1

# 10. Semáforo controlado por un pulsador

- Entrada: pulsador
- Salida: 2 LEDs
- Objetivo: cambiar LED activo al accionar el pulsador



# 11. Bonus: Acelerómetro



View pin map

I2C module: 0

Slave address: 10

Slave byte order: BigEndian

Enable register access

Slave register address: 0

Data type: uint8

Data size (N): 1

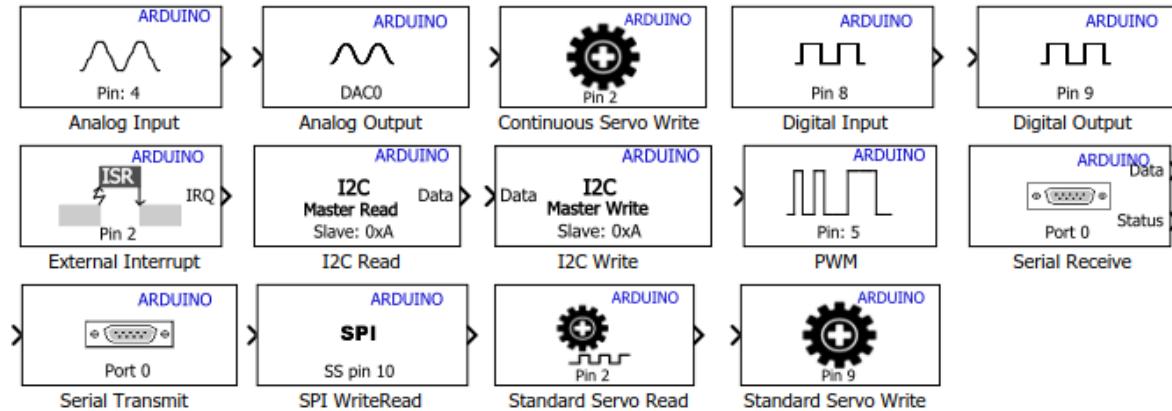
Output error status

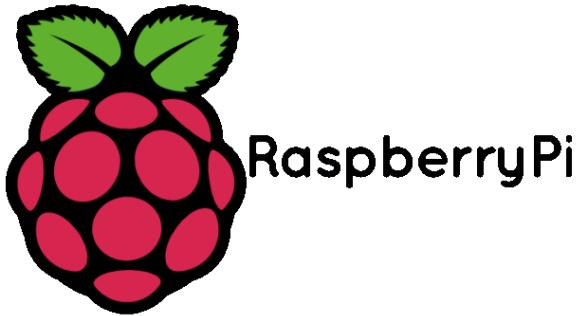
Sample time: 0.1

- Acelerómetro GY 521 (triaxial & giróscopo)
- Comunicación I2C
- En Arduino Uno
  - A4 → SCL
  - A5 → SDA

# 11. Resumen

- Uso de “common blocks” Arduino
- Conexionado HW
- Selección de pines de acuerdo a la placa que tengamos
- Acondicionamiento y “tipado” de los datos
- Tiempo de muestreo importante
- Simplicidad
- Otros no vistos hoy: I2C, TCP/IP, ...





# Módulo 3: Programación de Raspberry Pi con Simulink.

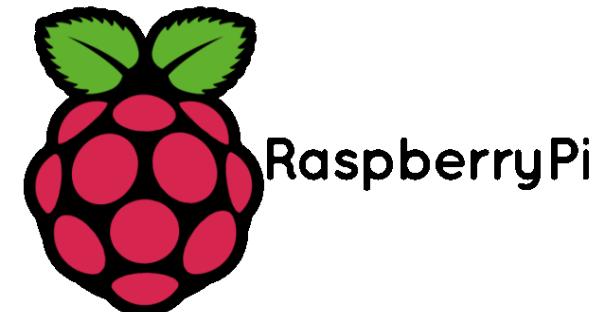
- a) Placas Raspberry PI. Descripción.  
Funcionamiento
- b) Proyecto de Raspberry PI usando  
Simulink.

# 1. Raspberry PI

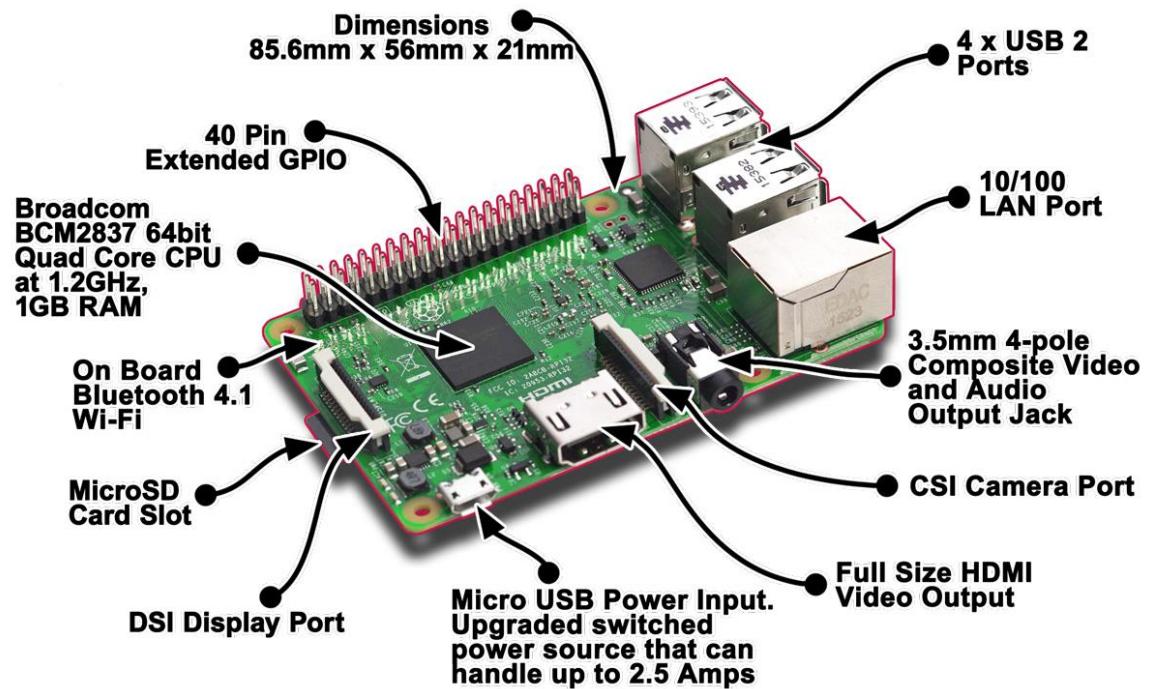
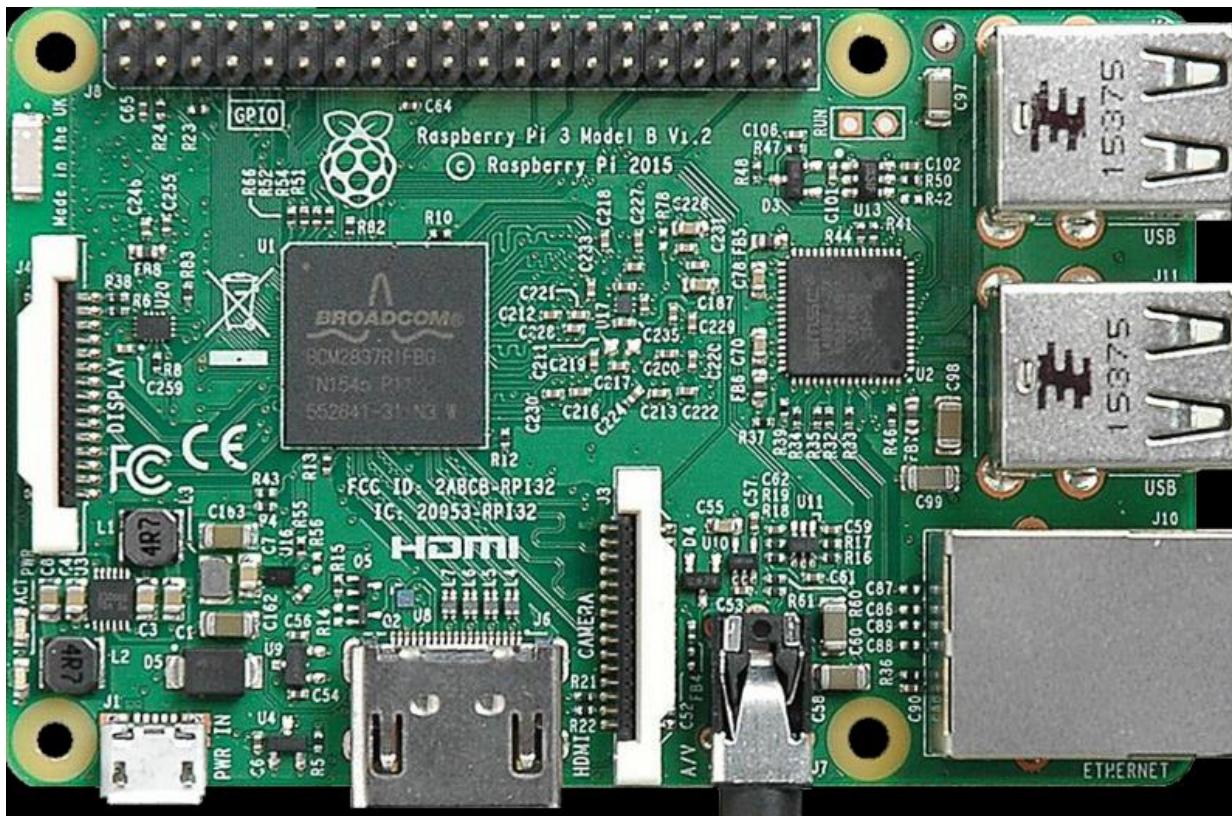
- PC de bajo coste desarrollado por la Raspberry Foundation.
- Uso libre bajo determinados supuestos (educacional, etc.).
- Admite varios sistemas operativos (Linux y... Windows).
- Dispone de un procesador, memoria RAM, una GPU, puertos USB, HDMI, Ethernet, WiFi, ...
- Gran versatilidad y potencia.
- Bajo coste: 40-50 €.

# 1. Raspberry PI 3 Model B+

- Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz
- 1GB LPDDR2 SDRAM
- 2.4GHz and 5GHz IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth 4.2, BLE
- Gigabit Ethernet sobre USB 2.0 (maximum throughput 300 Mbps)
- 40-pin GPIO
- HDMI
- 4 puertos USB 2.0
- Puerto de cámara CSI
- Puerto DSI (touchscreen)
- Audio stereo & video compuesto
- Puerto microSD para datos y sistema operativo
- 5V/2.5A DC



# 1. Raspberry Pi

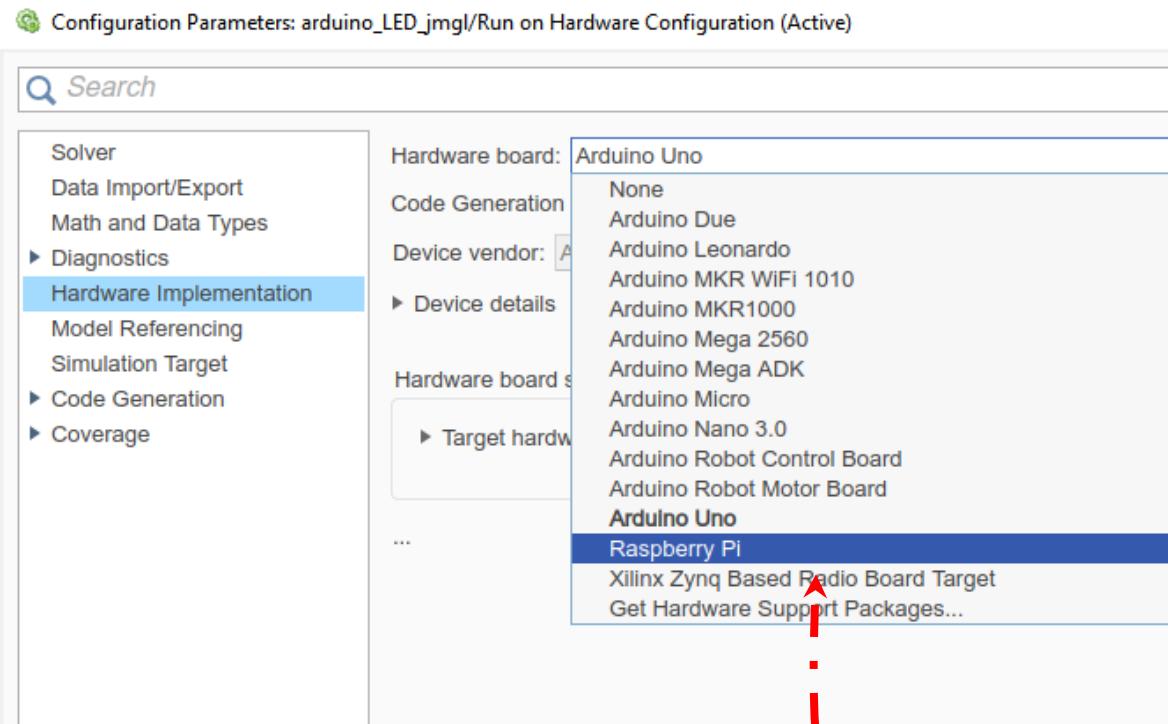


## 2. Instalación y configuración de Raspberry Pi en MATLAB & Simulink

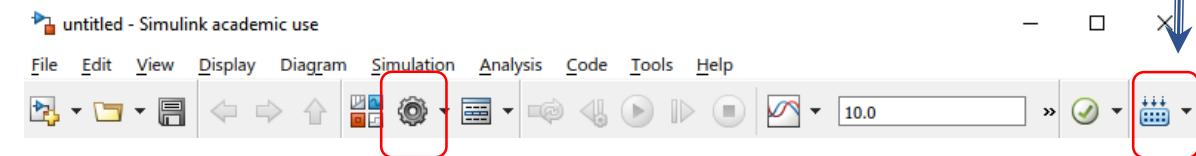
- Proceso similar al del “Support package para Arduino”
  - Descarga
  - Instalación
  - Configuración
- Diferencia notable: MATLAB & Simulink requieren que en la microSD haya una versión modificada del sistema operativo, de forma que sea compatible
- Necesario cargar imagen (se descarga de github)



# 2. Raspberry Pi en Simulink



- Es necesario ‘direccional’ el dispositivo.
- El código que tengamos (modelo) lo cargamos en Simulink pulsando en ‘deploy to hardware’.
- Posibilidad de trabajar de forma análoga a Arduino (GPIO)
- Creación de ejecutables Linux en la Raspberry (en su directorio raíz)
  - Archivo ELF



# 2. Raspberry PI en Simulink

- Pasos:
  - Identificar el HW
  - Direccionarlo (IP, login & password)
  - Deploy to hardware
- Lanzamos y verificamos el modelo desde la CLI de MATLAB:

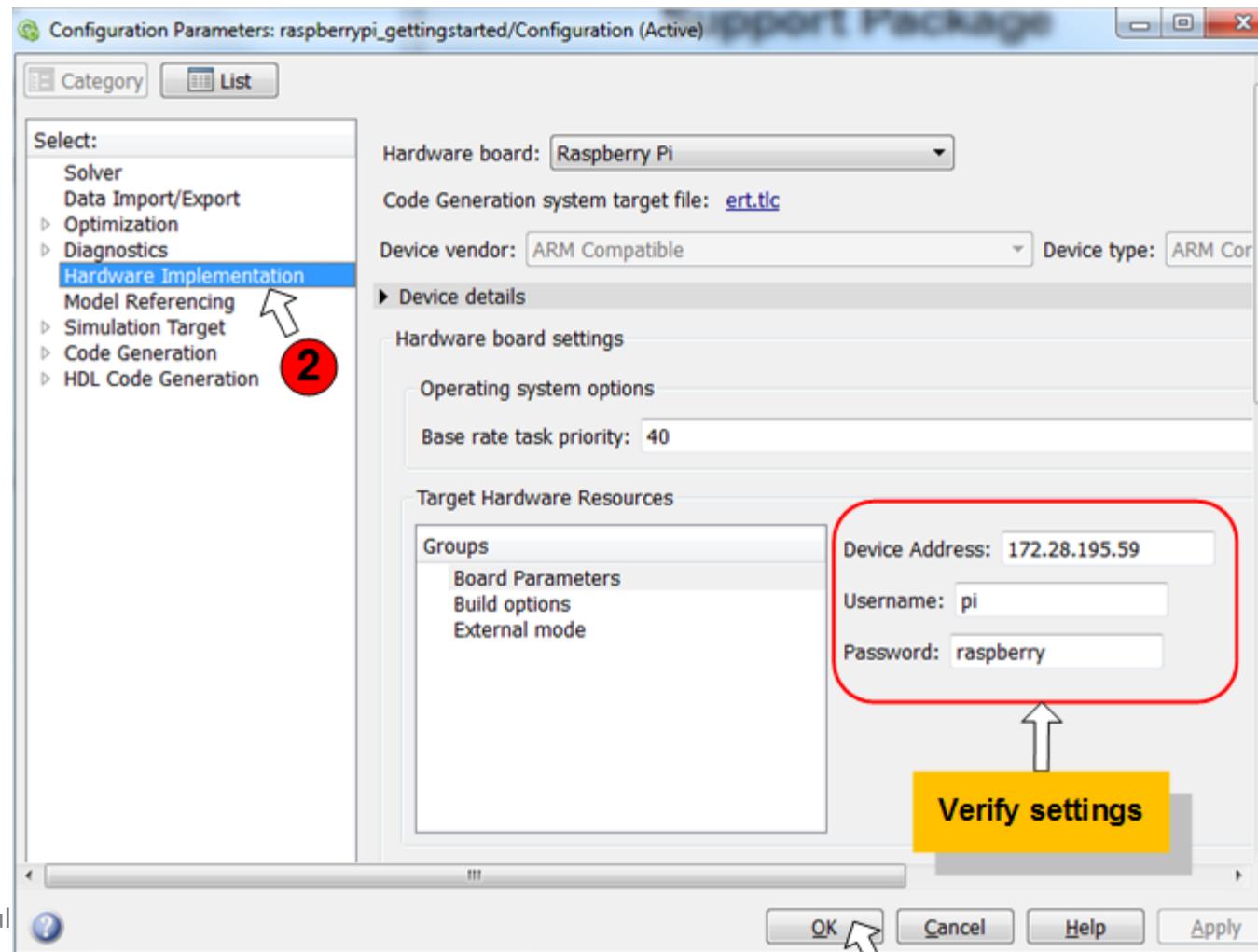
```
>> r = raspberrypi; %objeto Raspberry
```

```
>> isModelRunning(r,'miModelo')
```

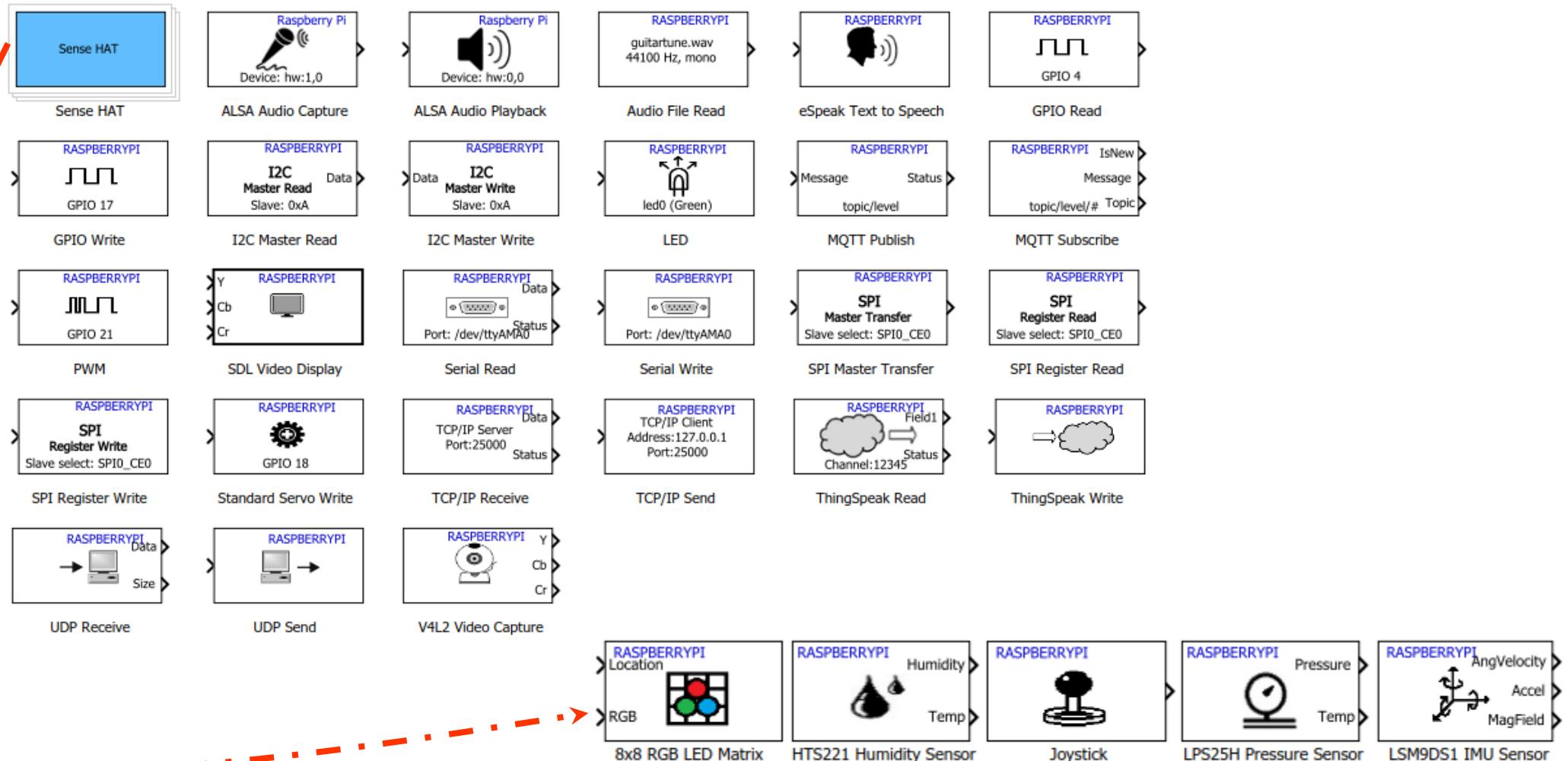
```
>> runModel(r, 'miModelo')
```

```
>> stopModel(r,'miModelo')
```

- Permite funcionamiento autónomo de la RasPI

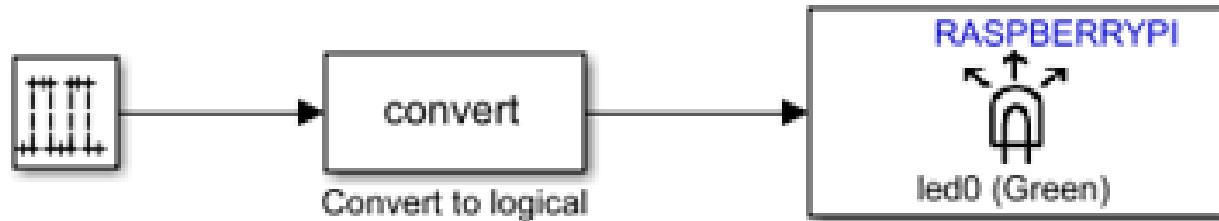


## 2. Raspberry Pi en Simulink



### 3. LED intermitente en Raspberry PI

- Misma filosofía que con Arduino
- Ejemplo preconfigurado en el “support package” de Simulink  
`>> open_system('raspberrypi_gettingstarted_unconfigured.slx')`



# Bonus: Control de una radio definida por software con Simulink.

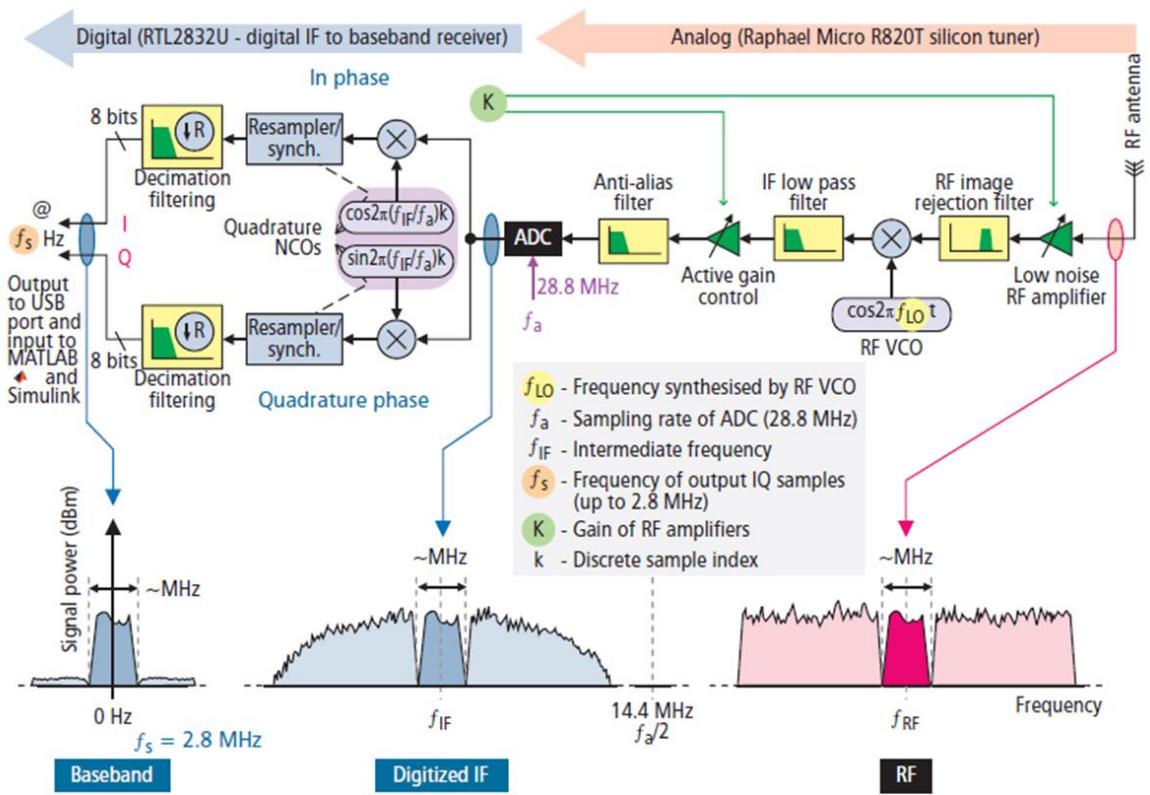
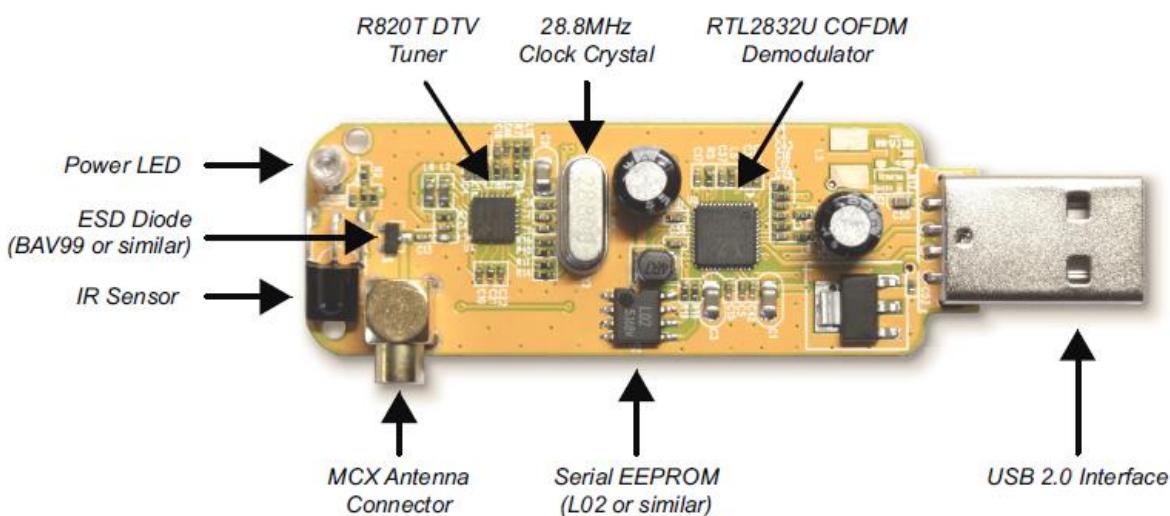
- a) RTL-SDR. Concepto
- b) Hardware y software
- c) Ejemplo de un analizador de espectro con una RTL-SDR



# 1. RTL-SDR

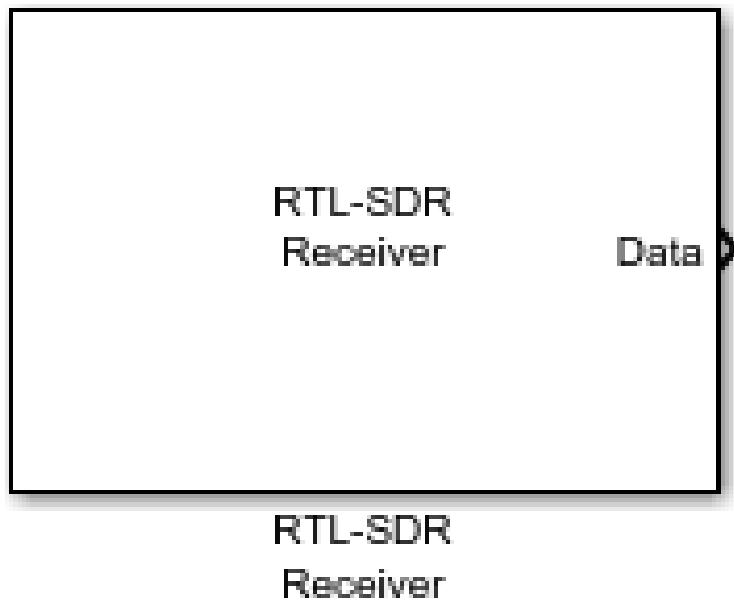
- Dongle USB inicialmente pensado para recibir señales de TDT (DVB-T).
- Se vio que podría trabajar como un receptor de propósito general: software-defined radio (SDR).
  - Flexibilidad en la funcionalidad: solo cambios SW (vía Simulink)
- Bajo coste: 15 €.
- Programable con MATLAB & Simulink, GNURadio, LabView, ...
- Comunidad en Internet muy activa.
- Rango de frecuencias: 25-1700 MHz.
- ADC: 8 bits.

# 1. RTL-SDR: hardware & software



Imágenes: R. W. Stewart, K. W. Barlee, D. S. W. Atkinson, and L. H. Crockett, Software Defined Radio using MATLAB & Simulink and the RTL-SDR, Published by Strathclyde Academic Media, 2015 (Softback ISBN 9780992978716, Hardback ISBN 9780992978723).

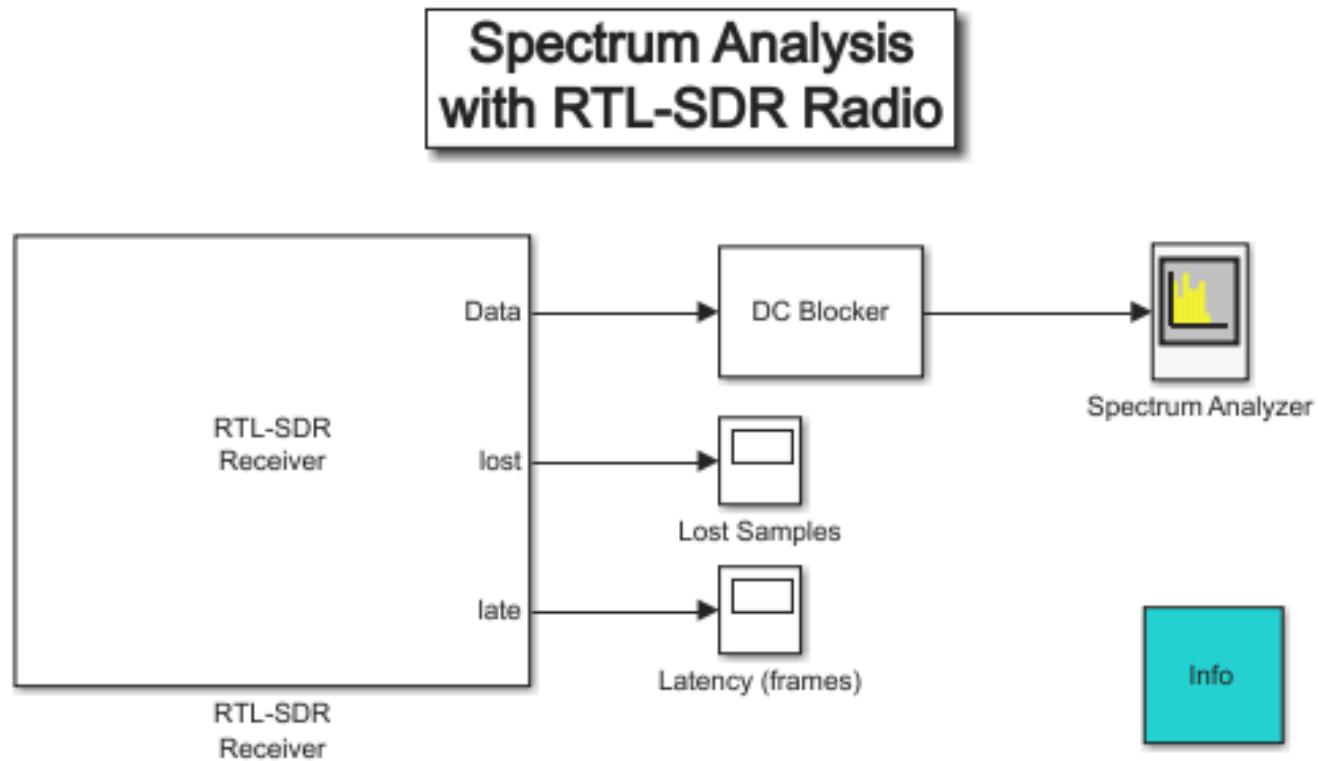
## 2. RTL-SDR en Simulink



- En Simulink trabajamos con un bloque que encapsula el RTL.
- La salida (SW) es el puerto 'Data' de la imagen (emula el USB).
  - 8 bits I/Q
  - $f_s$
  - $f_c$
- MATLAB soporta otras radios software de mayor complejidad como Pluto, USRP, etc.



### 3. Analizador de espectro con RTL-SDR en Simulink



# Introducción a Simulink con Arduino y Raspberry Pi

Juan Moreno García-Loygorri

Madrid, 20-21/MAY/2019