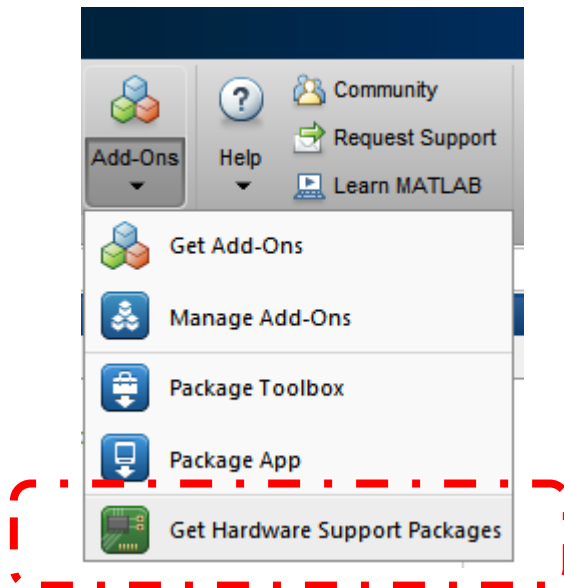


Introducción a Simulink con Arduino y Raspberry Pi

Juan Moreno García-Loygorri
Madrid, 20-21/MAY/2019

9. Instalación y configuración de Arduino en MATLAB & Simulink

- Instalación

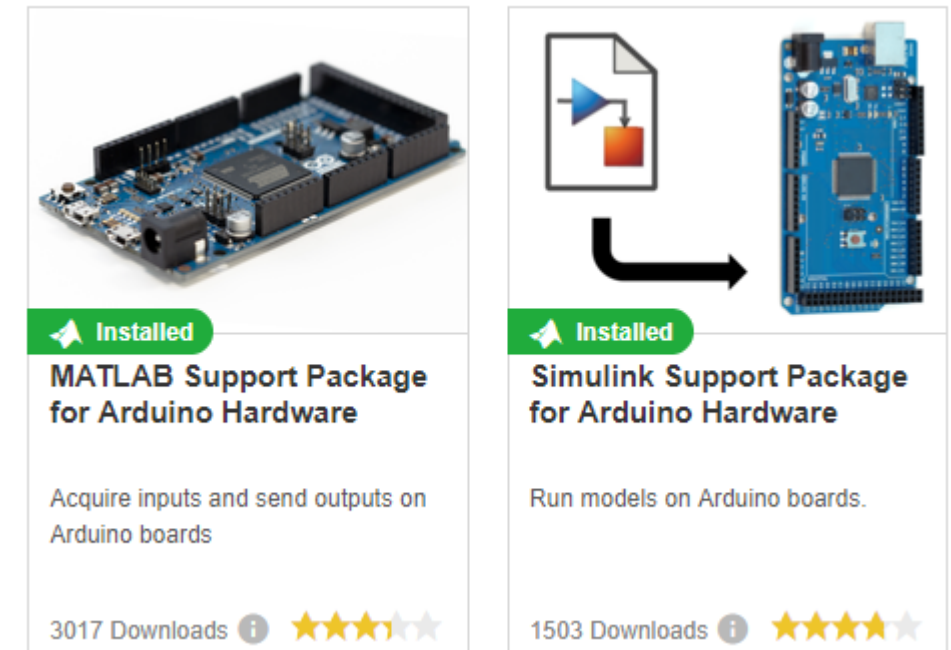


Si se instala el de Arduino, se genera una dependencia con el de MATLAB y se instalarán ambos

Pasos:

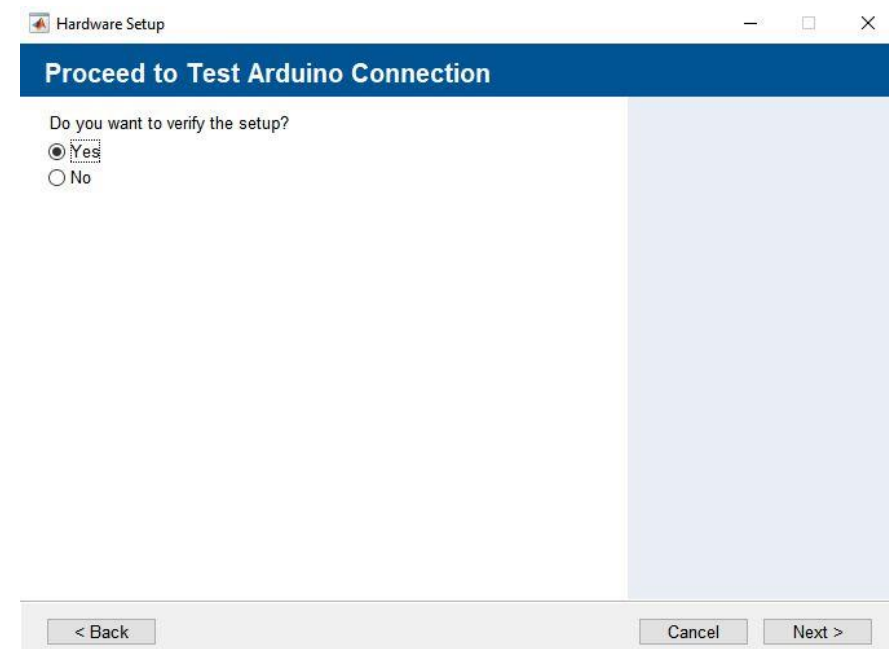
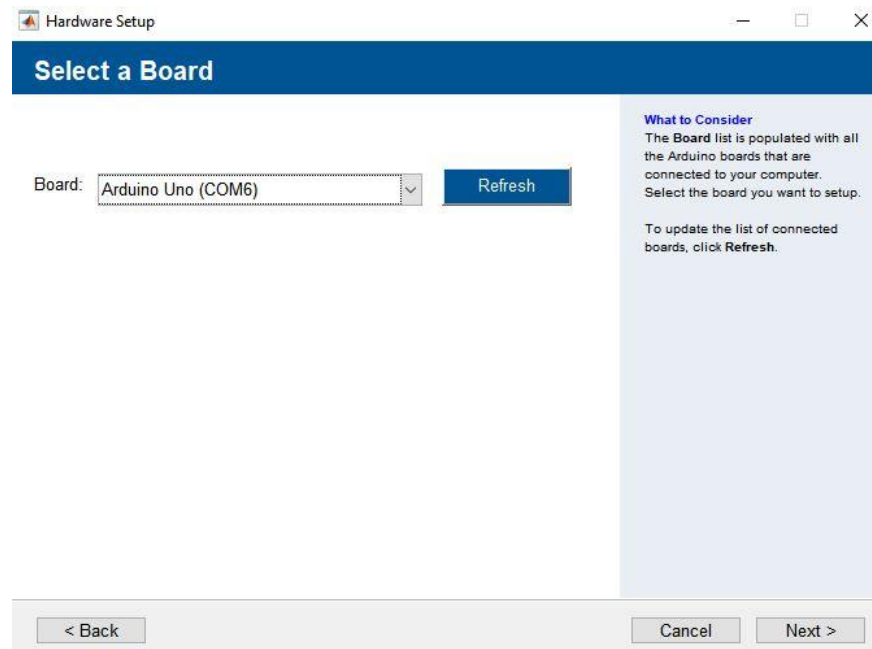
- Descarga
- Instalación
- Configuración (se requiere el HW)

Hardware Support Packages (304)



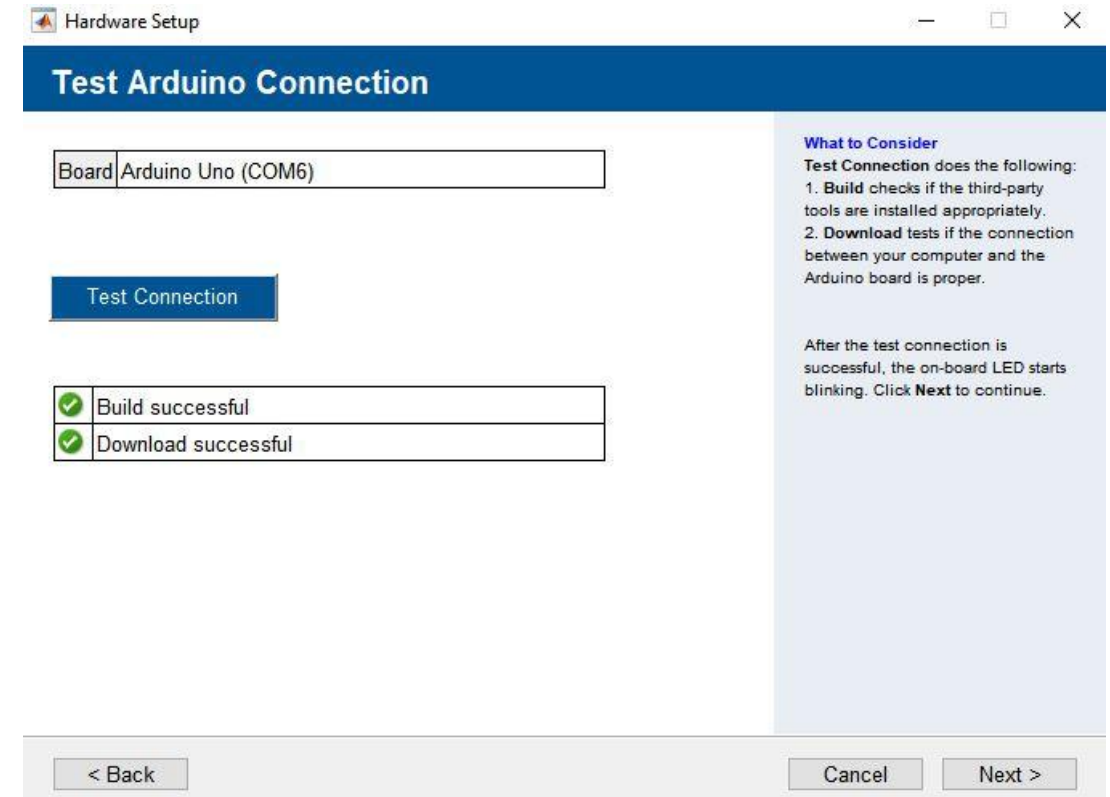
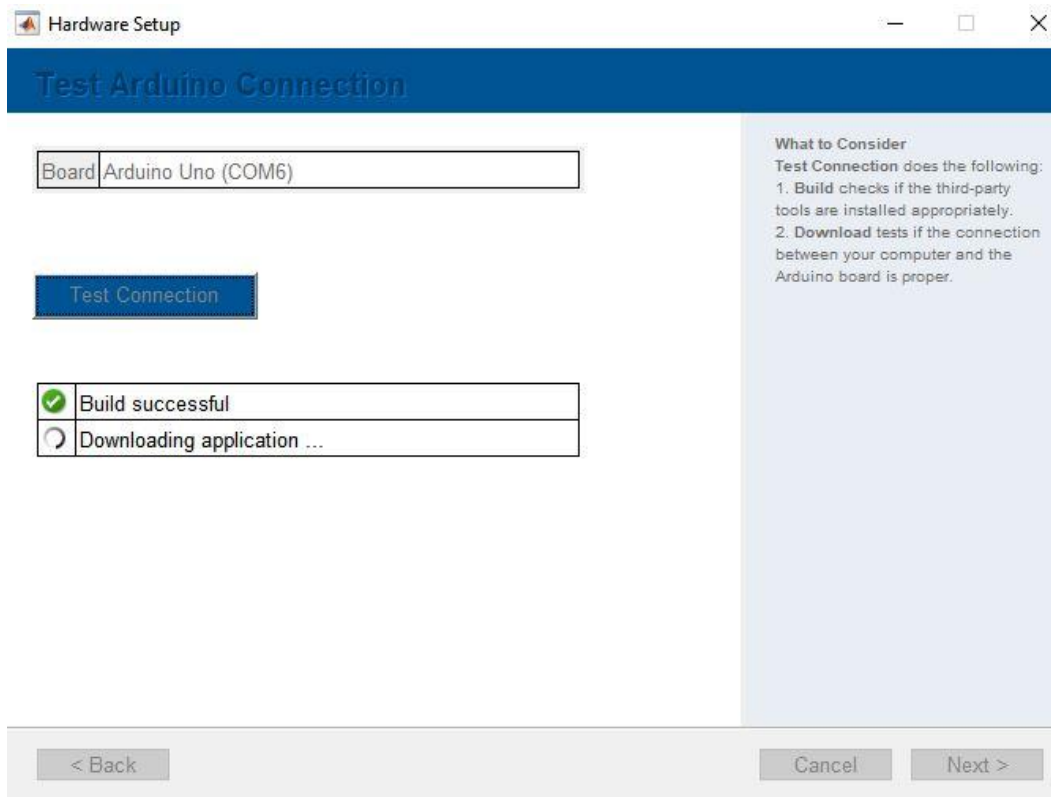
9. Instalación y configuración de Arduino en MATLAB & Simulink

- Configuración tras la instalación



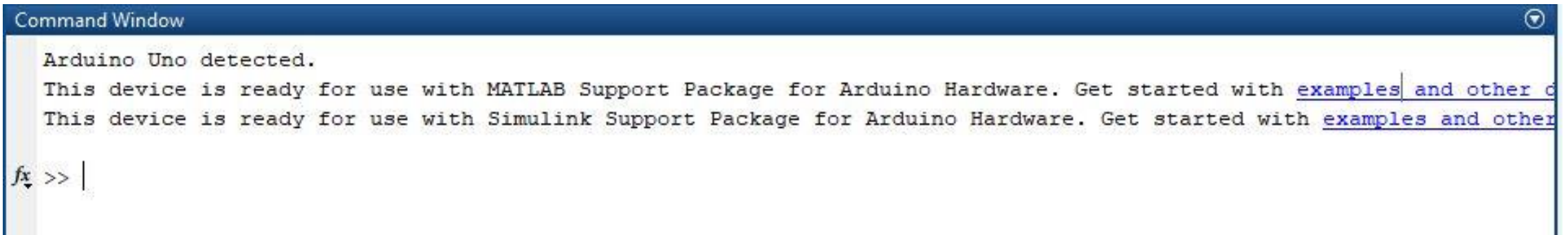
9. Instalación y configuración de Arduino en MATLAB & Simulink

- 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



```
Command Window

Arduino Uno detected.
This device is ready for use with MATLAB Support Package for Arduino Hardware. Get started with examples and other c
This device is ready for use with Simulink Support Package for Arduino Hardware. Get started with examples and other

fx >> |
```

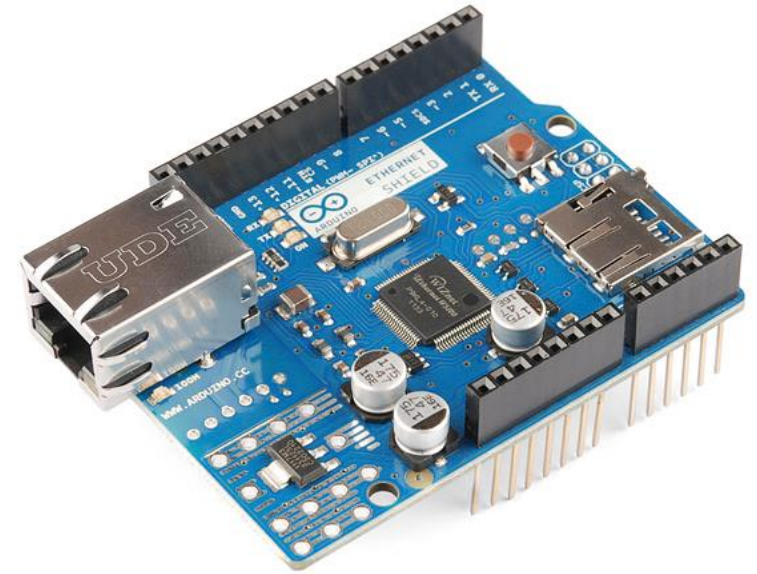
- Potencial problema con puerto COM. Resolución:
 - En Simulink: Simulation → Configuration parameters → Hardware implementation
 - Board → comprobar que seleccionamos nuestra placa Arduino
 - Target HW resources → seleccionar manualmente el puerto que corresponda

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
 - Importación/Exportación a WS MATLAB
- Salvado/importación de disco
- Facilidad para sistemas dinámicos
- Atención con tiempo de simulación y *sample time*

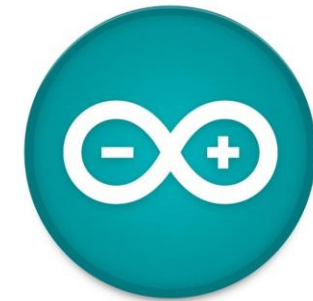
Módulo 3: Programación de Arduinos con Simulink.

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.



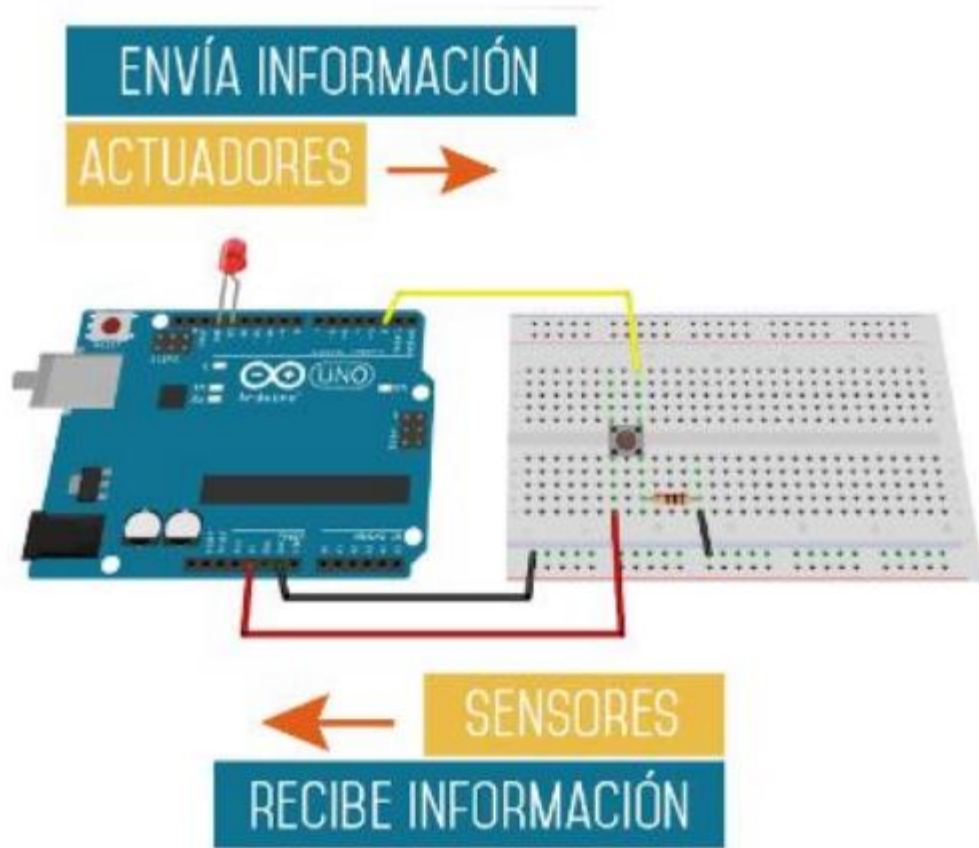
1. Arduino

- Familia de placas de desarrollo hardware para sensorización y control.
- Basados en microcontroladores (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 correc
}

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

    printRAW();

    delay(100);
}
```

1. Arduino

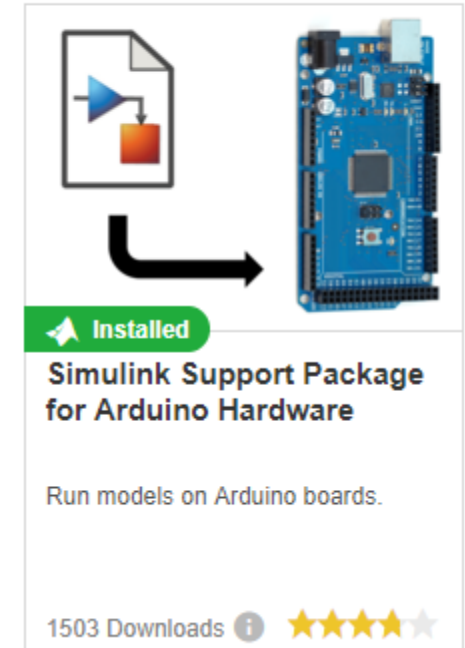
- El IDE de Arduino es el framework más utilizado y más potente
- El principal problema que tiene es la curva de aprendizaje → código
- Carga del SW, depuración, etc., son relativamente asequibles
- Ayuda en la web:
 - Gran disponibilidad de ejemplos
 - Comunidad de usuarios muy activa



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)

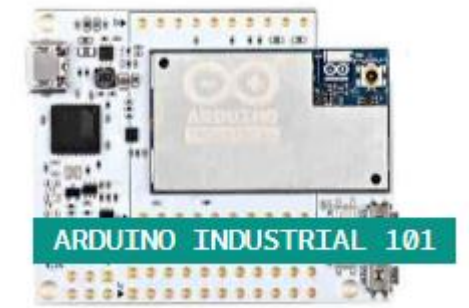
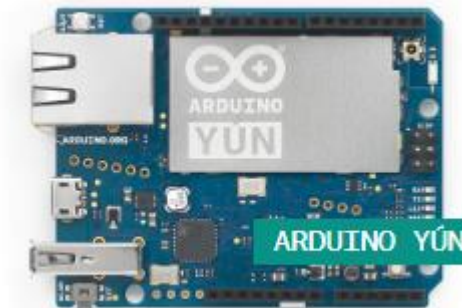


1. Arduino

- 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)

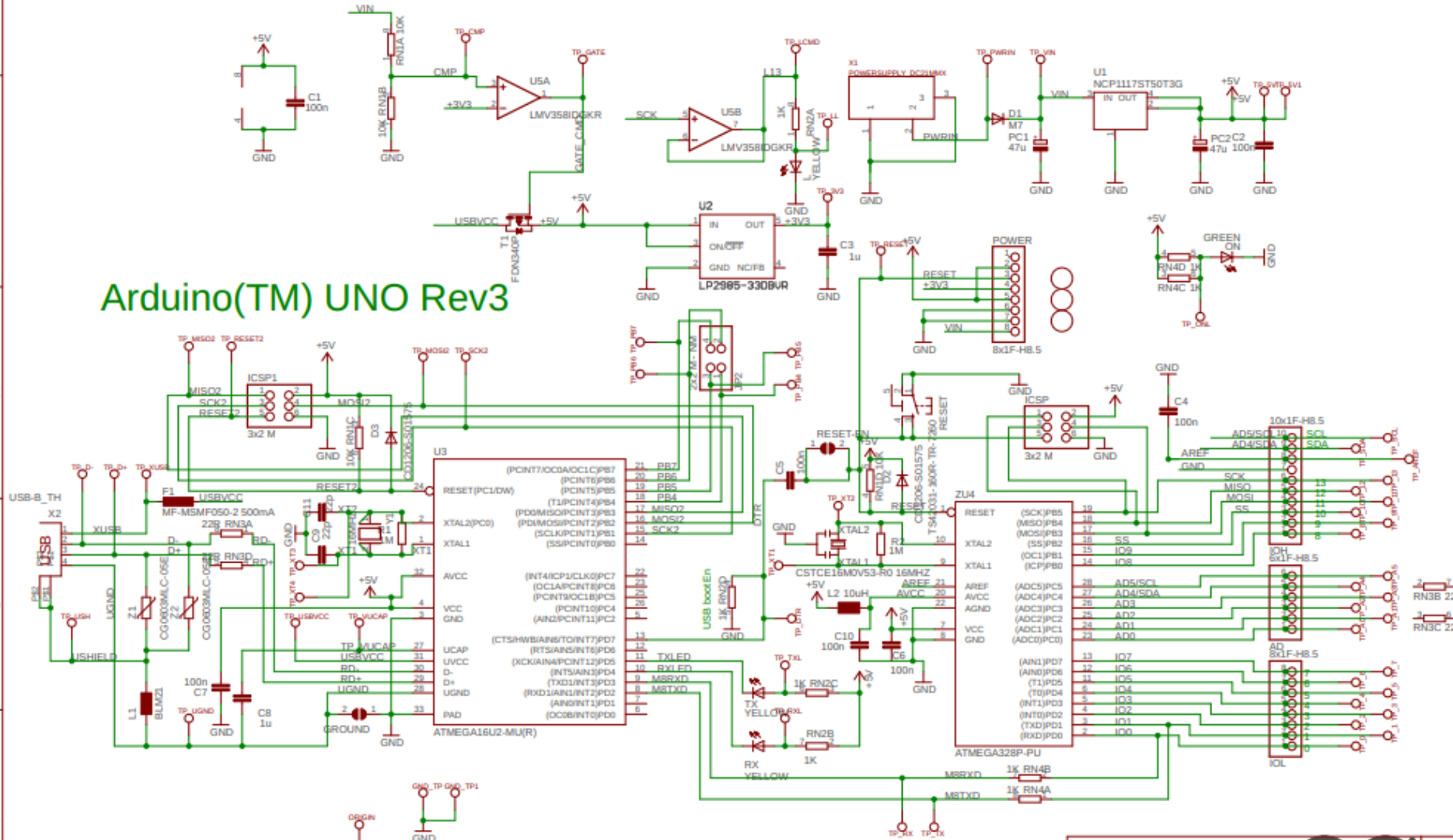


1. Arduino Uno

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

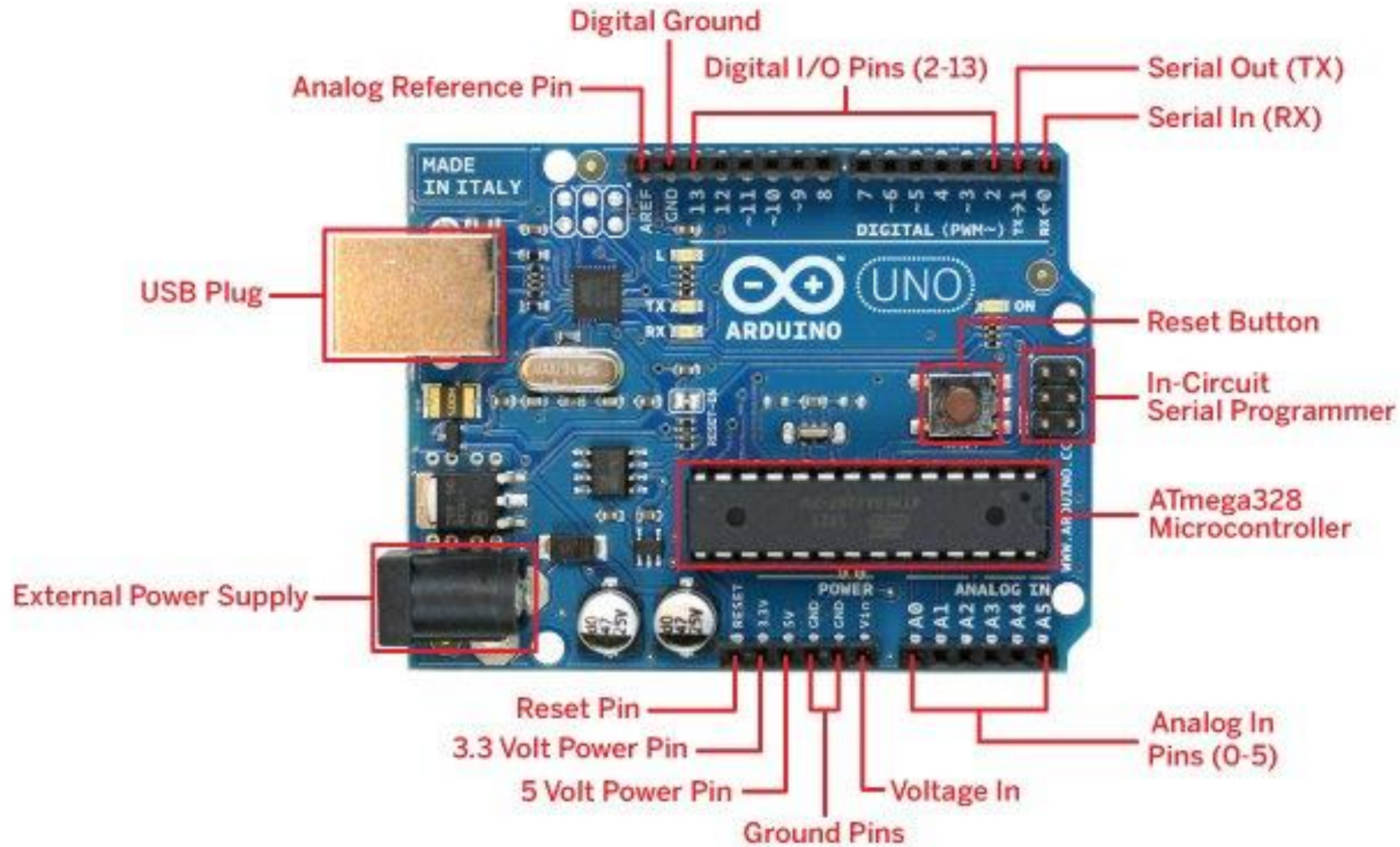
- Modelo más popular en entorno educativo
- Lo usaremos en este laboratorio
- Existen alternativas no Arduino (i.e. Elegoo Uno) 😊
- 14 pines digitales (6 PWM)
- 6 pines analógicos
- @16 MHz
- EEPROM de 1K
- Protección sobrecorrientes (>500 mA) en puerto USB

Arduino(TM) UNO Rev3



Reference Designs ARE PROVIDED "AS IS" AND "WITH ALL FAULTS". Arduino SA DISCLAIMS ALL OTHER WARRANTIES, EXPRESS OR IMPLIED, REGARDING PRODUCT, INCLUDING BUT NOT LIMITED TO, ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Arduino SA may make changes to specifications and product descriptions at any time, without notice. The Customer must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Arduino SA reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The product information on the Web Site or Materials is subject to change without notice. Do not finalize a design with this info. ARDUINO and other Arduino brands and logos and Trademarks of Arduino SA. All Arduino SA Trademarks cannot be used without owner's formal permission.





1. Arduino

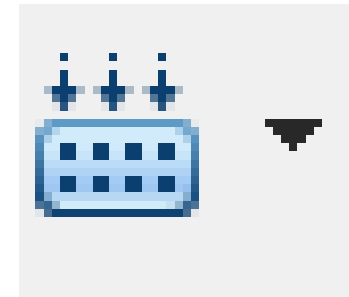
- Los arduino NO son sistemas de tiempo real
 - (*) Hay algunas iniciativas para hacerlo, como ARTe
- Importante tener en cuenta los tiempos de cada tarea que llevamos a cabo.
- IDE Arduino tiene mejores métodos para monitorizar esto
- (Mucho prueba y error)

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

Commands		Duration		
		µs	CPU cycles	
Various Arduino commands	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

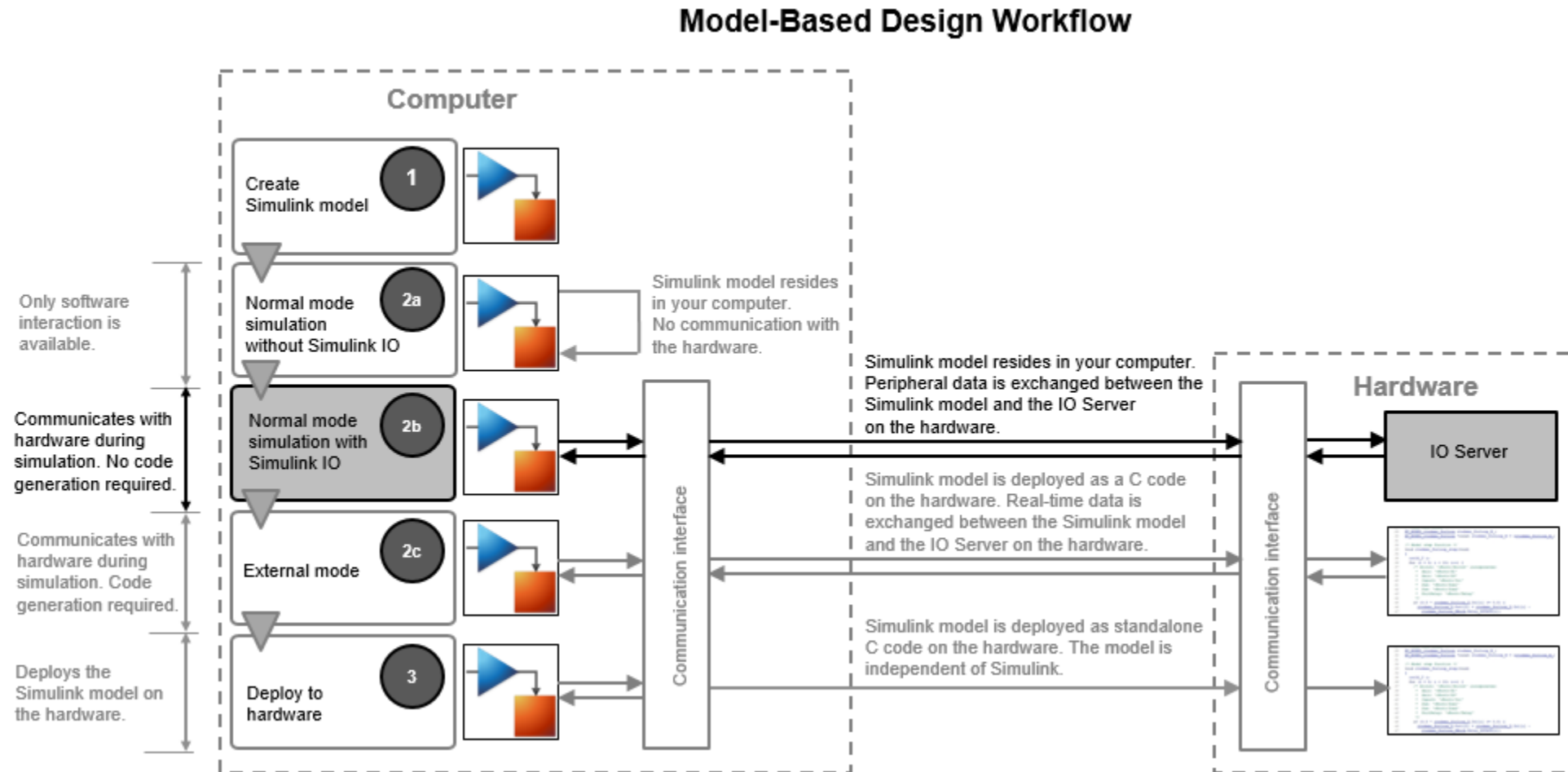
- Haremos nuestro modelo en Simulink.
 - Definición de pines
 - Lógica para actuadores / sensores
- Carga en dispositivo → “Deploy to hardware” (en vez de “Run”, como en las simulaciones)
- Fases de la carga:
 - Compiling...
 - Initializing...
 - Building...
 - Ready



Buena política: simular primero (“Run”)

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

2. Arduino en Simulink (modelo HW en general)



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
 - ...

Code Generation Report

Find: Match Case

Contents

- [Summary](#)
- [Subsystem Report](#)
- [Code Interface Report](#)

Generated Code

- [-] Main file
 - [ert_main.c](#)
- [-] Model files
 - [arduino_gettingstarted.c](#)
 - [arduino_gettingstarted.h](#)
 - [arduino_gettingstarted_private.h](#)
 - [arduino_gettingstarted_types.h](#)
- [-] Data files
 - [arduino_gettingstarted_data.c](#)
- [+] Utility files (2)
- [+] Interface files (1)
- [+] Other files (6)

Code Generation Report for 'arduino_gettingstarted'

Model Information

Author	The MathWorks, Inc.
Last Modified By	The MathWorks, Inc.
Model Version	1.84
Tasking Mode	MultiTasking

[Configuration settings at time of code generation](#)

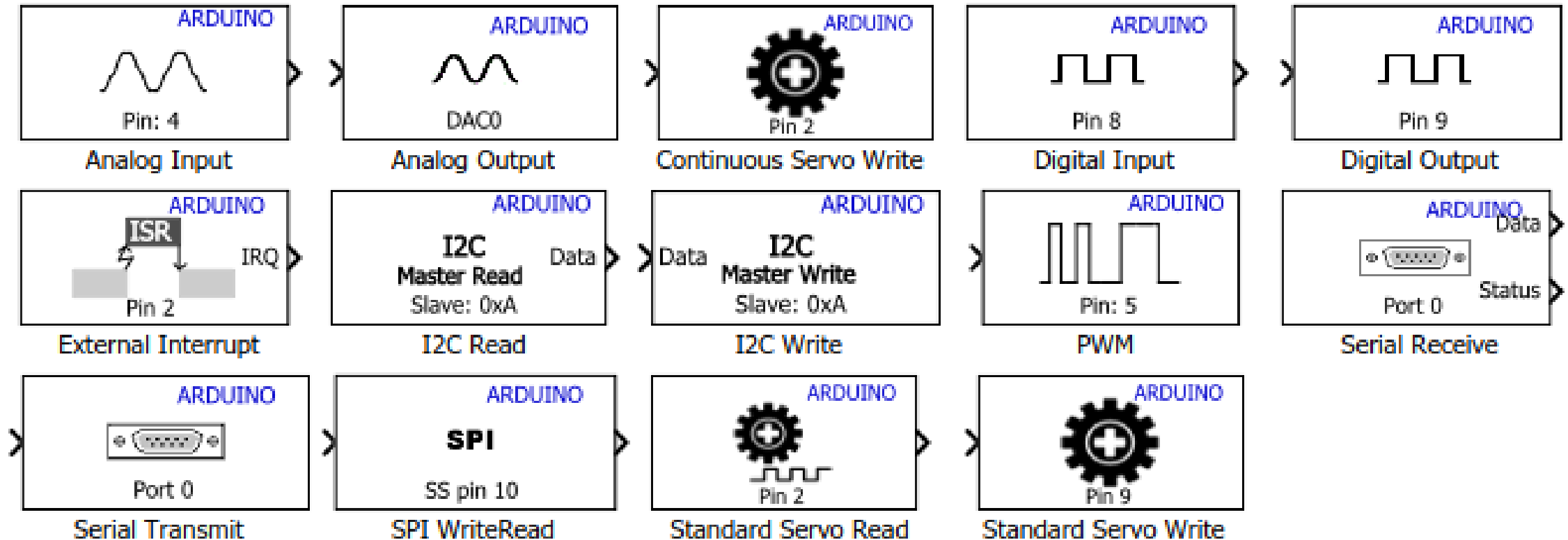
Code Information

System Target File	ert.tlc
Hardware Device Type	Atmel->AVR
Simulink Coder Version	9.1 (R2019a) 23-Nov-2018
Timestamp of Generated Source Code	Mon May 20 01:29:36 2019
Location of Generated Source Code	C:\Users\P18543\Documents\Dropbox\workspace\arduino\simulink\arduino_gettingstarted_ert_rtw\Code
Type of Build	Model
Objectives Specified	Unspecified

Additional Information

Code Generation Advisor	Not run
-------------------------	---------

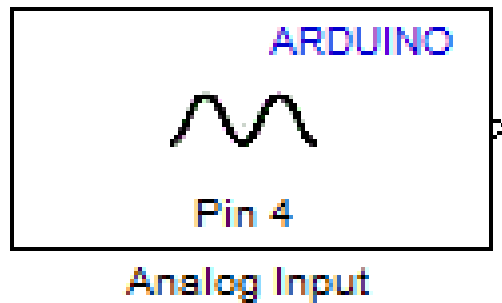
2. Arduino en Simulink. Common blocks



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

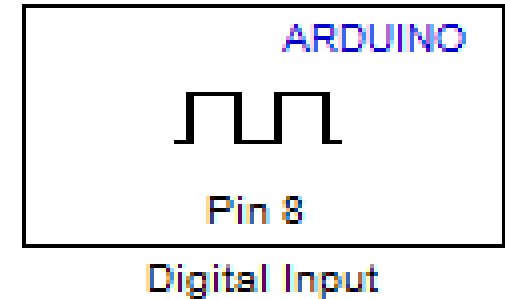
2. Arduino en Simulink

Analog 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

• Digital input

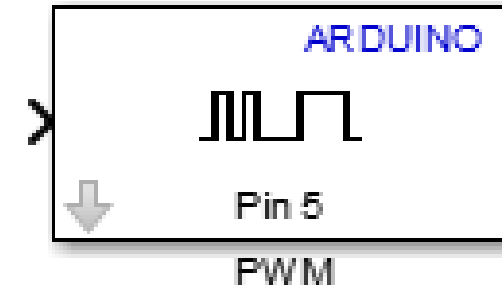
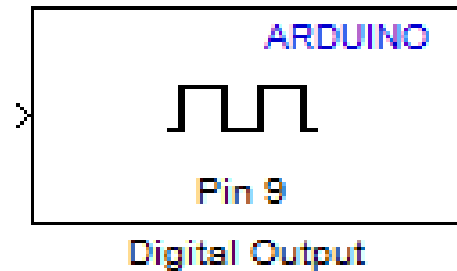


- Obtiene el valor LOGICO de un pin digital (0 V = '0'; 5 V = '1').
Boolean

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

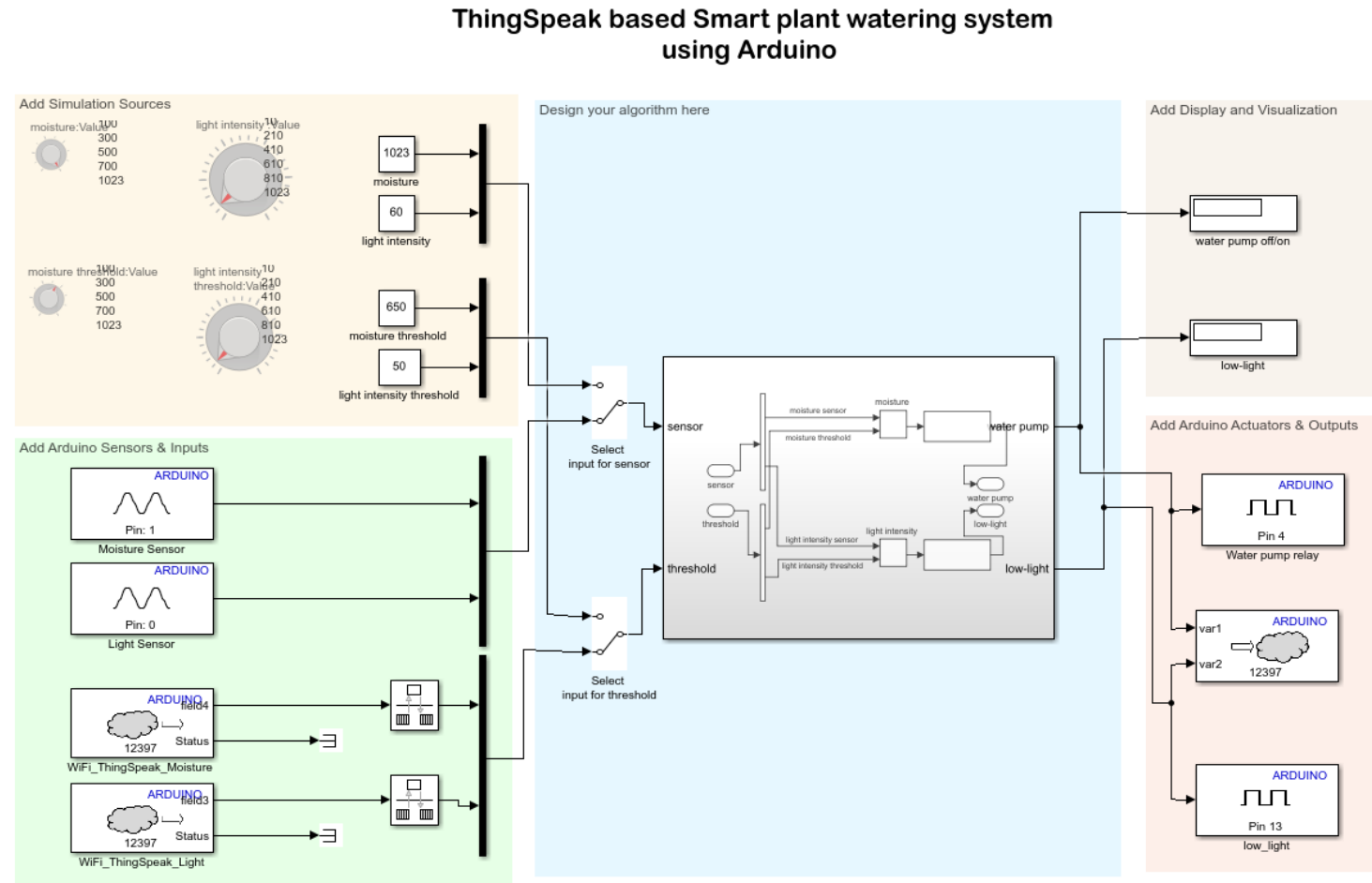
2. Arduino en Simulink

- 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_jmgl' 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
```

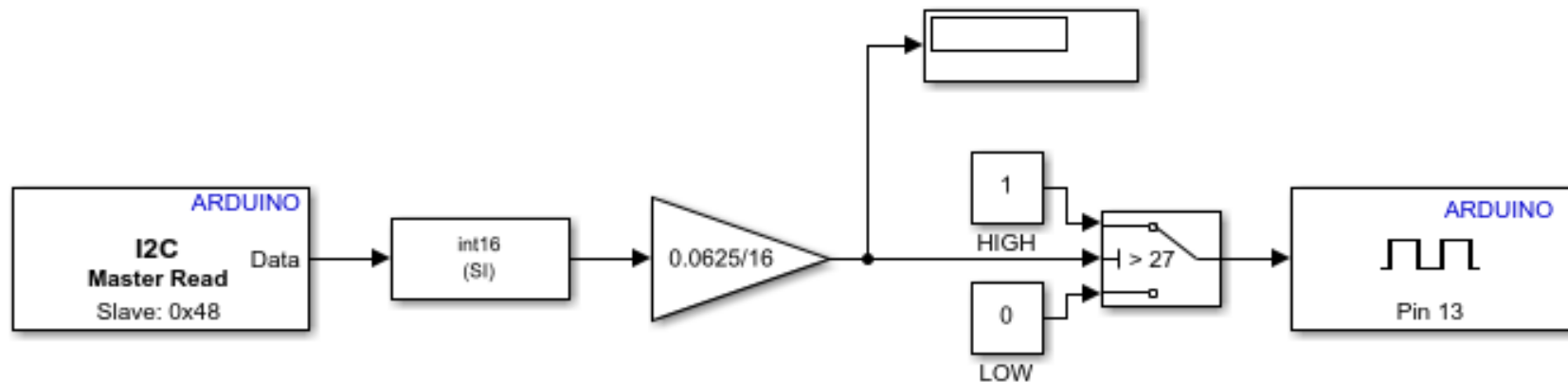
2. Arduino en Simulink. Filosofía de trabajo

Ejemplo para riego
de plantas



2. Arduino en Simulink. Filosofía de trabajo

Read temperature from an I²C 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 K Ω
- Consultar pines disponibles
- Bloque Digital output
 - Valor de salida es boolean
- Source: pulse generator
 - Control del “ciclo de trabajo”

3. Iluminación de un LED

 Block Parameters: Digital Output ×

Arduino Digital Output (mask) (link)

Set the logical value of a specified digital output pin.

Enter the number of the digital output pin. Do not assign the same pin number to multiple blocks within a model.

[View pin map](#)

Parameters

Pin number:

Parameters

Pulse type:

Time (t):

Amplitude:

Period (number of samples):

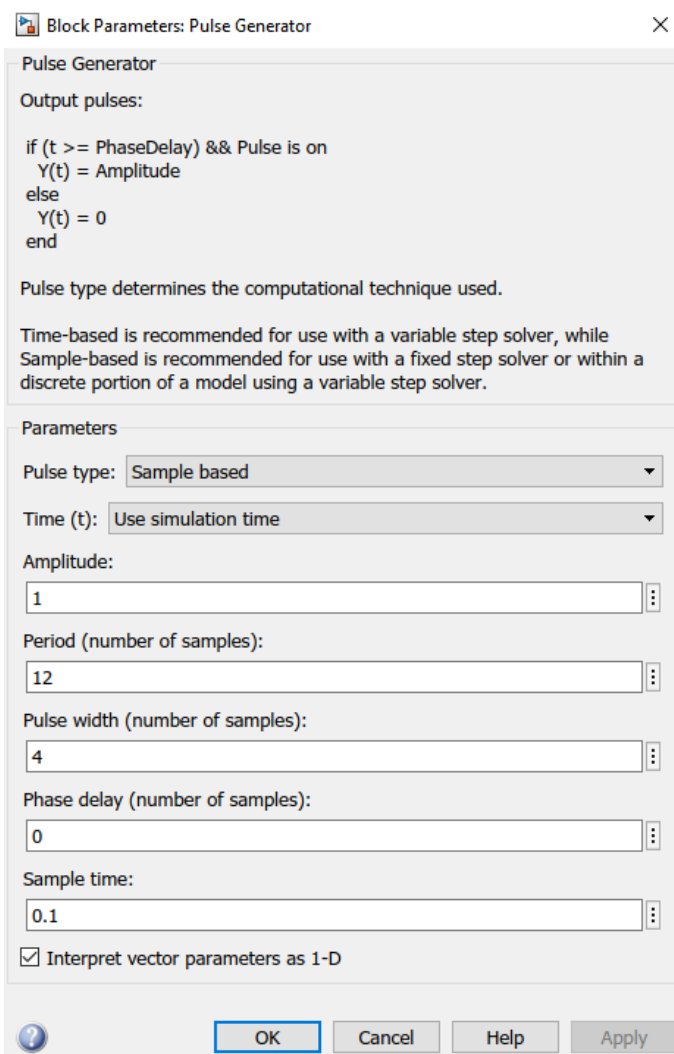
Pulse width (number of samples):

Phase delay (number of samples):

Sample time:

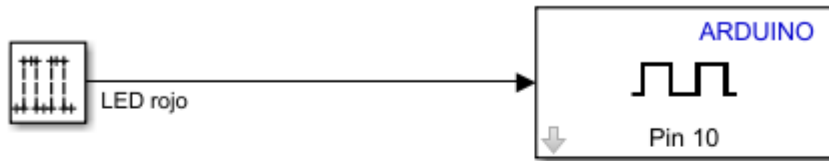
	Blocks		
		Uno	Nano 3.0
User Selectable Pins	Digital Input	0-13	0-13
	Digital Output	0-13	0-13
	Analog Input	0-5	0-7
	Analog Output	N/A	N/A
	PWM	3,5,6,9,10,11	3,5,6,9,10,11
	Standard Servo Read	0-13	0-13
	Standard Servo Write	0-13	0-13
	Continuous Servo Write	0-13	0-13
	External Interrupt	2,3	2,3
	SPI Slave Select (SS)	0-10	N/A
Fixed Pins	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

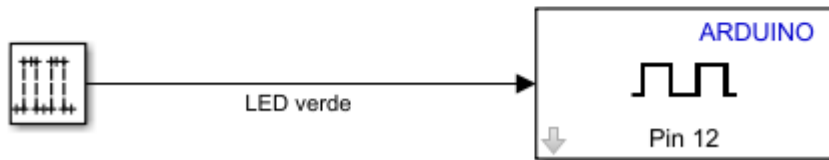


- Parámetros:
 - Periodo
 - Ancho del pulso
 - Desfase
 - Tiempo de muestreo
- Probemos a bajar el Ts al mínimo posible

5. Varios LEDs

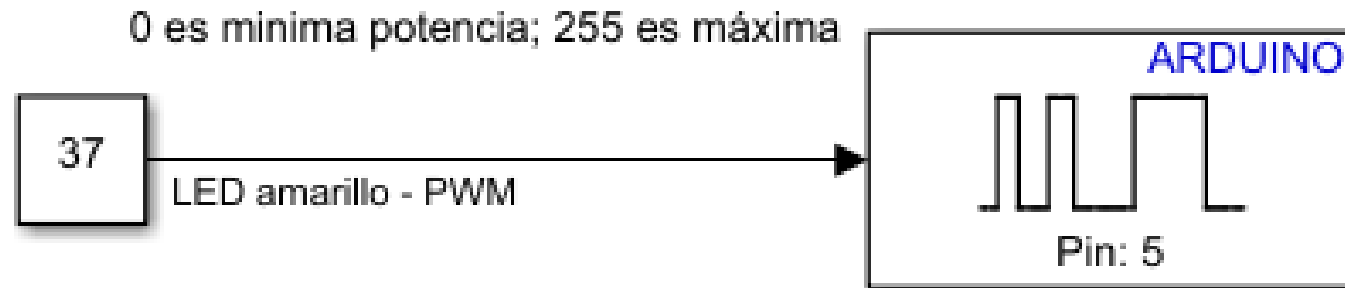


Ejemplo 2: LED. Añadir otro LED en pin 12



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

6. Ajuste de Intensidad de un LED

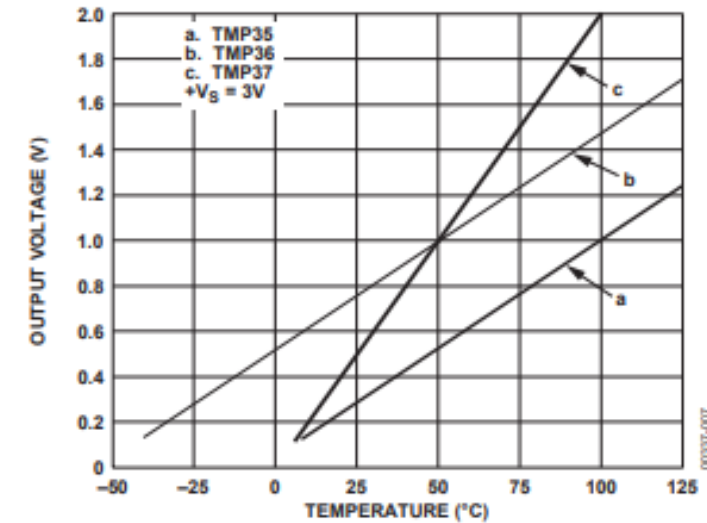
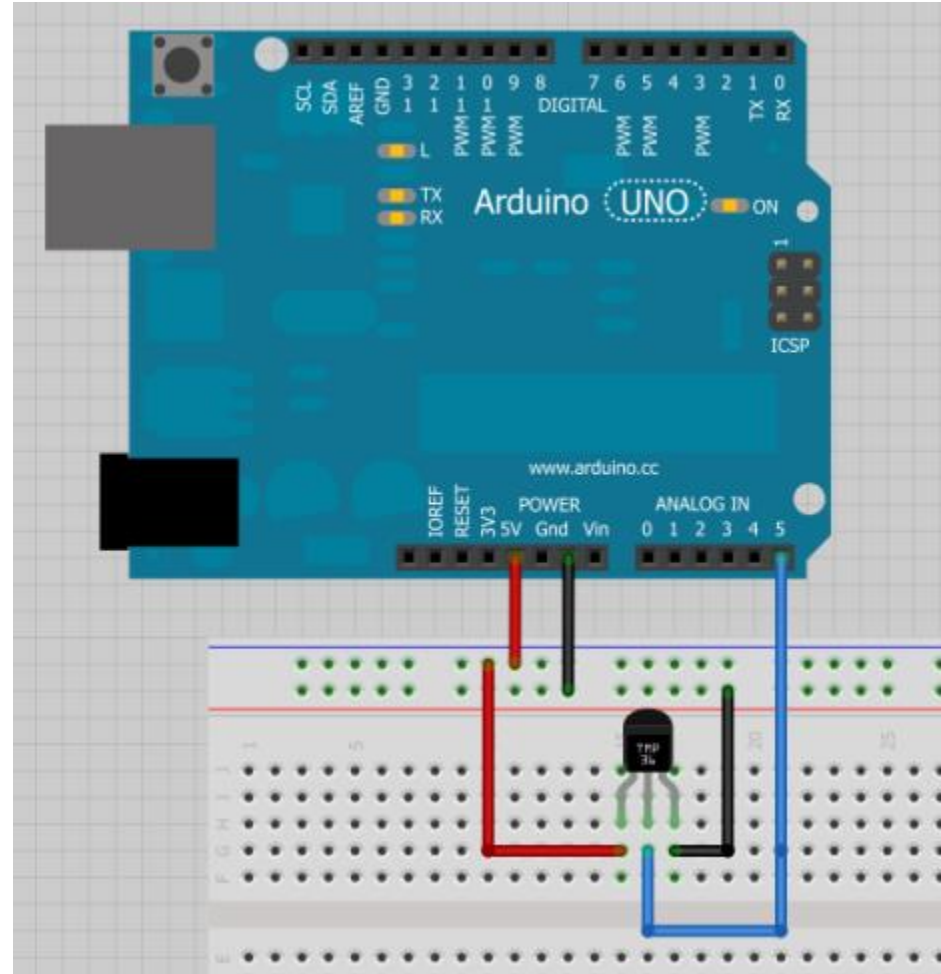


- Modulación por ancho de pulso
- Entrada entre 0-255, proporcional a la intensidad
- Importante: solo determinados puertos admiten PWM (ver hoja)

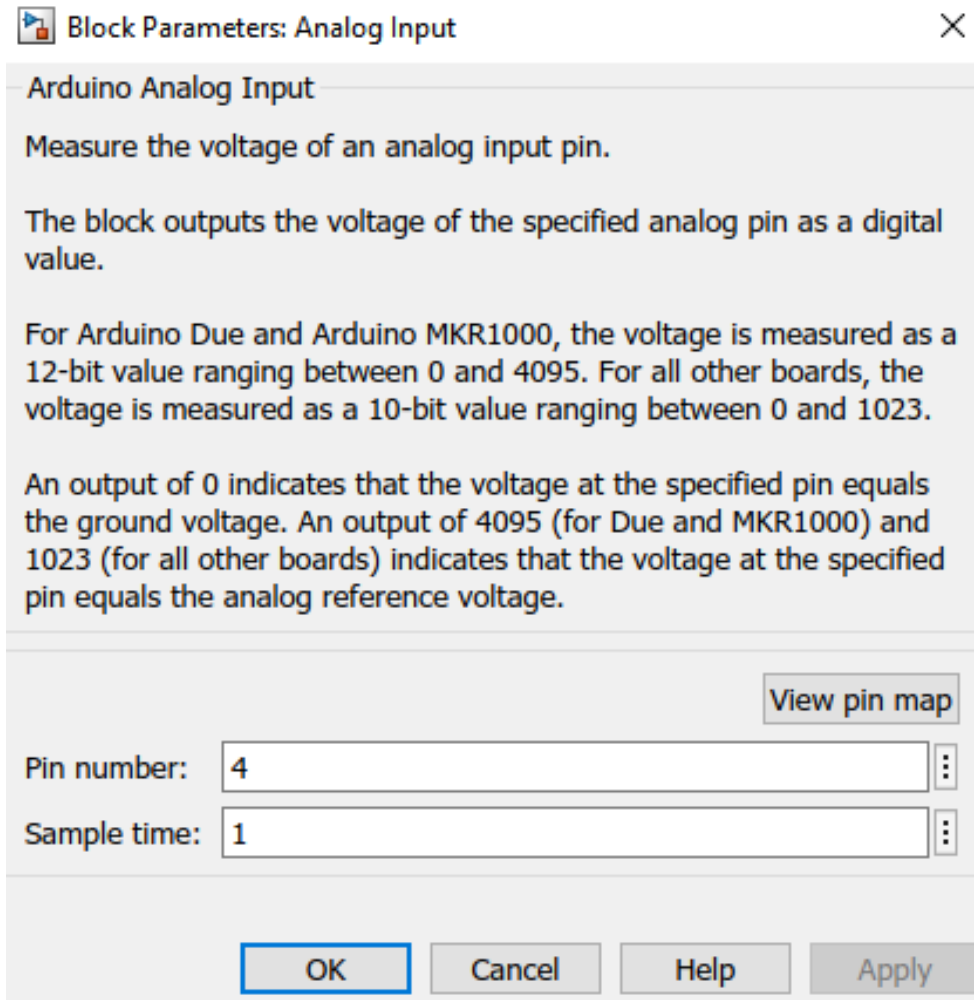
User Selectable Pins	Blocks	Uno
	Digital Input	0-13
	Digital Output	0-13
	Analog Input	0-5
	Analog Output	N/A
	PWM	3,5,6,9,10,11
	Standard Servo Read	0-13

7. Leer de un sensor analógico (TMP36)

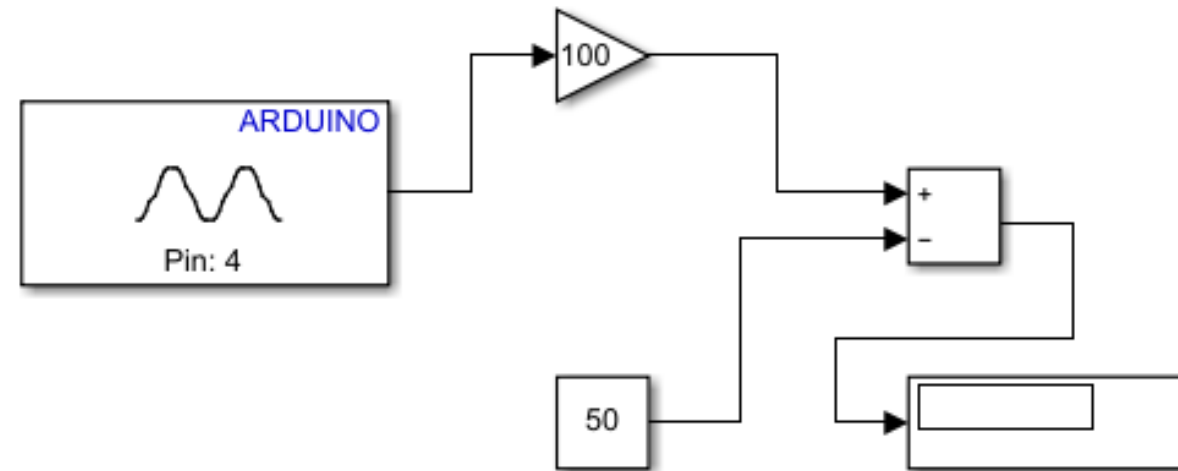
- Ejemplo analógico:
- Prestamos atención al pin que corresponda (son varios posibles)
- “procesado” valor leído para obtener medida
 - Ver hoja características sensor



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

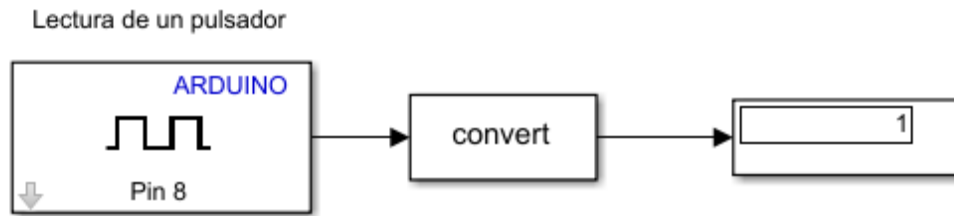
- Bonus: lectura con MATLAB

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

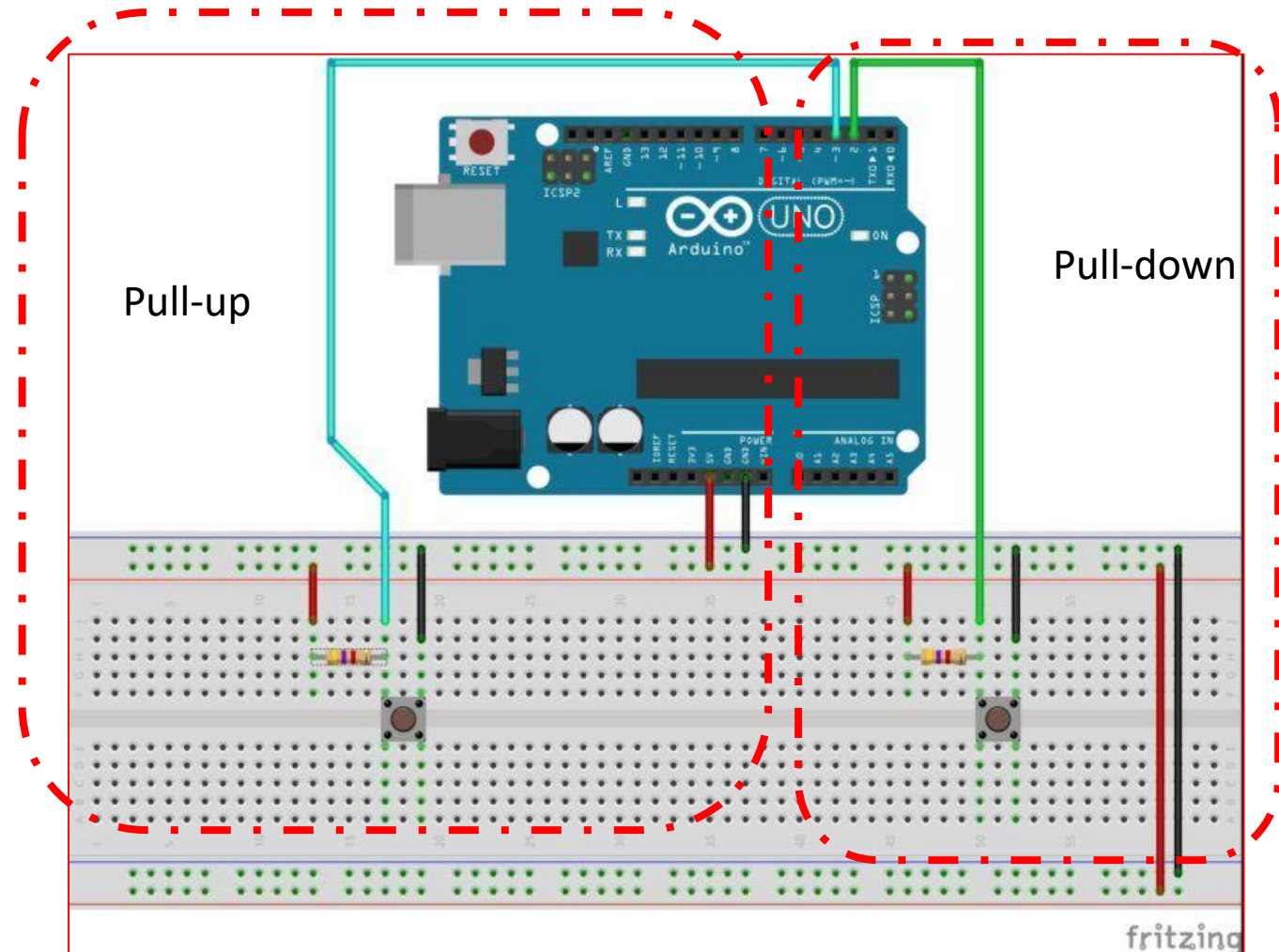
- Problemática potencial en gestión de la comunicación (o solo Simulink o solo MATLAB)
 - Se soluciona borrando el objeto arduino ('a') antes de usarlo en otro sitio

8. Trabajar con un pulsador

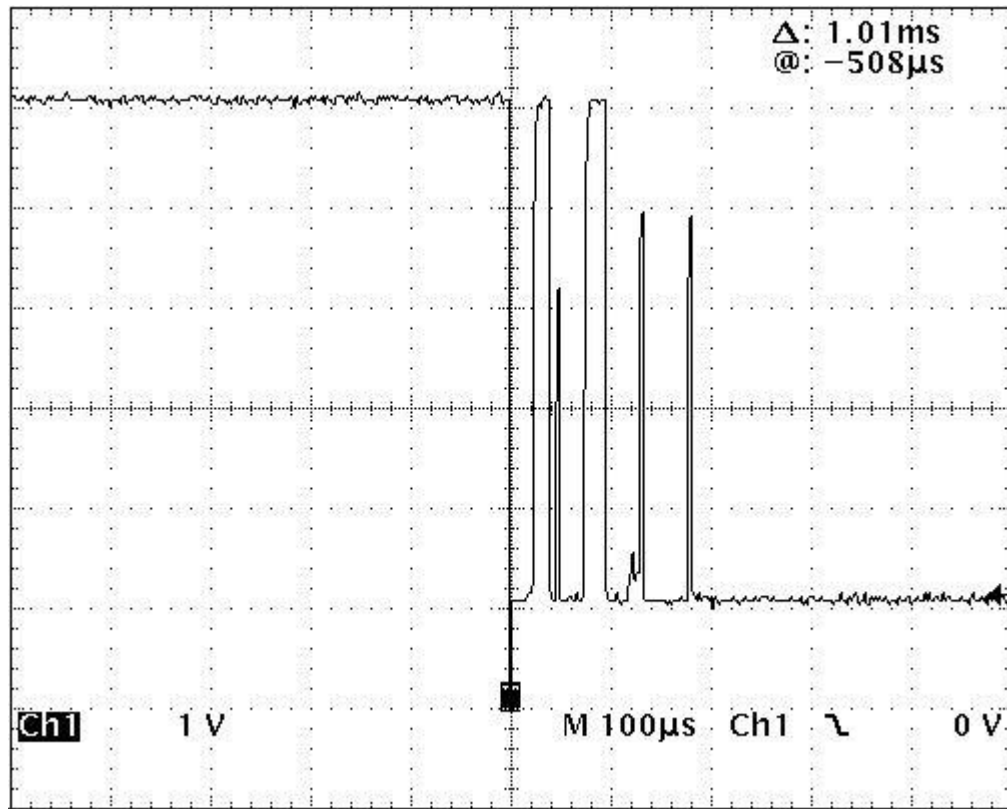
- Resistencia de pull-up (o down) necesaria



- 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

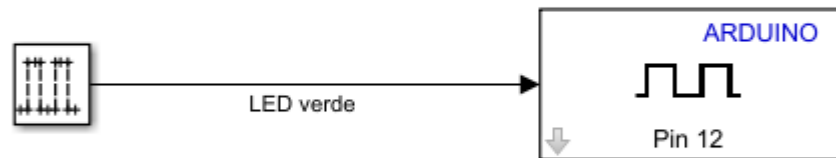
- Poner 3 LED, uno de cada color cada uno en una salida digital de Arduino
- Con “Pulse Generator”, asignar subperiodos dentro de un ciclo a cada LED, de forma que solo brille uno.
- Se puede ajustar la intensidad de cada uno.
- Sin gestión de “exclusividad”



9. Semáforo cíclico



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



Este LED tiene un desfase de 8 Ts con respecto al rojo



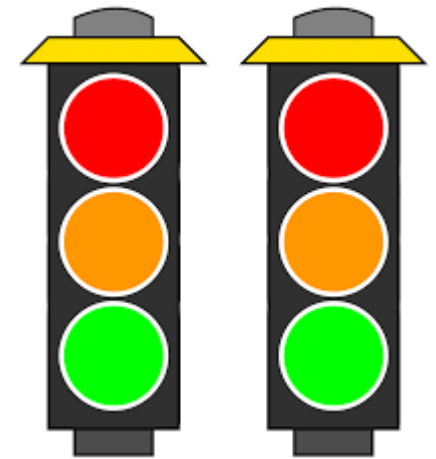
Parameters
Pulse type: <input type="text" value="Sample based"/>
Time (t): <input type="text" value="Use simulation time"/>
Amplitude: <input type="text" value="1"/>
Period (number of samples): <input type="text" value="12"/>
Pulse width (number of samples): <input type="text" value="4"/>
Phase delay (number of samples): <input type="text" value="0"/>
Sample time: <input type="text" value="0.1"/>

Parameters
Pulse type: <input type="text" value="Sample based"/>
Time (t): <input type="text" value="Use simulation time"/>
Amplitude: <input type="text" value="1"/>
Period (number of samples): <input type="text" value="12"/>
Pulse width (number of samples): <input type="text" value="4"/>
Phase delay (number of samples): <input type="text" value="8"/>
Sample time: <input type="text" value="0.1"/>

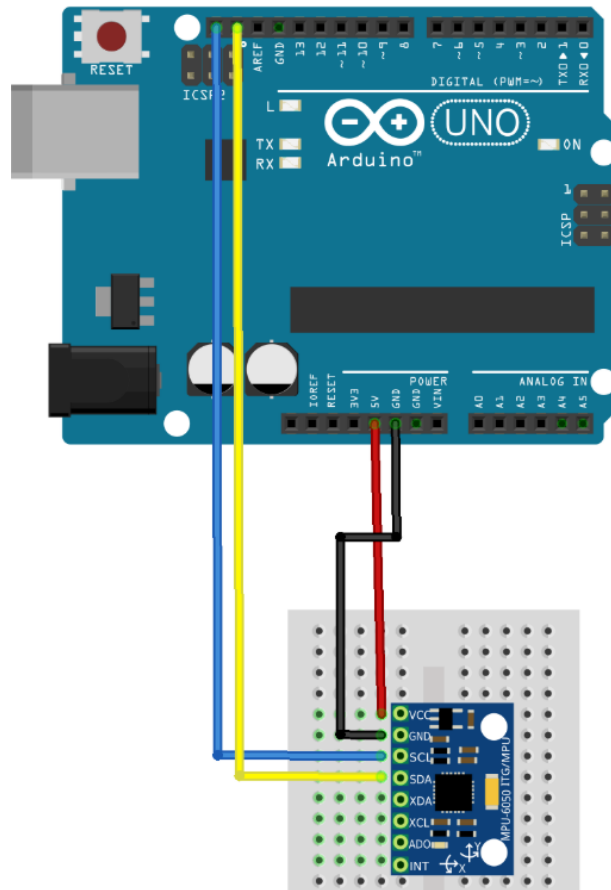
Parameters
Pulse type: <input type="text" value="Sample based"/>
Time (t): <input type="text" value="Use simulation time"/>
Amplitude: <input type="text" value="1"/>
Period (number of samples): <input type="text" value="12"/>
Pulse width (number of samples): <input type="text" value="4"/>
Phase delay (number of samples): <input type="text" value="4"/>
Sample time: <input type="text" value="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, ...

