PRÁCTICAS DE REGULACIÓN AUTOMÁTICA (IEEF ETSIDI-UPM)

Carlos Platero

Práctica 1: Introducción al laboratorio de Regulación Automática

Las prácticas de Regulación Automática se realizarán con Matlab y Arduino. Antes de iniciar el laboratorio, se deberá tener instalado la versión de Matlab con licencia UPM. Se utilizará el toolbox de Control System y Signal Processing. Utilizar el comando ***ver*** (en la línea de comandos) para verificar los paquetes instalados.

Así mismo, se instalará los paquetes de soporte de MATLAB y Simulink para Arduino:

https://es.mathworks.com/videos/series/arduino-with-matlab-and-simulink-99406.html

Habrá cuatro prácticas y un examen teórico-experimental que se realizará en la última sesión.

En cada sesión y al iniciar la práctica, los alumnos entregarán una memoria con la documentación del trabajo experimental de la anterior práctica y la resolución de las cuestiones teóricas planteadas en la práctica a realizar. La nota del laboratorio será una combinación de la calificación del examen de prácticas junto con las notas de las memorias entregadas.

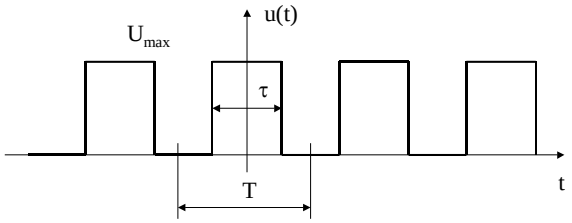
Material a emplear:

- Protoboard
- Polímetro
- Arduino (se recomienda Mega2560)
- 3 LM358N
- Resistencias: 10 MΩ, 150 kΩ, 33 kΩ, 68 kΩ, 680Ω
- Potenciómetro de 100kΩ, aunque puede ser de cualquier valor superior a 10kΩ
- Condensador: 2.2 µF 1 µF, 180 nF
- Led
- Cables
- Pinzas de cocodrilo para el polímetro
- Canal de video accesible para interaccionar con los montajes.

PRÁCTICA 1. INTRODUCCIÓN A MATLAB/SIMULINK ARDUINO

Series de Fourier de una señal

Dada una señal se procederá a obtener el espectro de Fourier. Observar que las señales que se puede introducir en Matlab son de tiempo finito y discreto, i.e. las señales se adquieren mediante muestras y utilizando una frecuencia de muestreo. En este ejemplo, se analiza una señal cuadrada en el rango de 0V a 5V con un periodo de 10 ms (ver apuntes de teoría, capítulo 2).



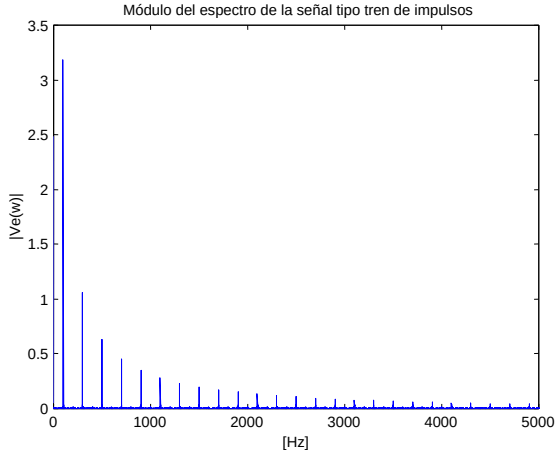
Como la señal es de tipo par, sólo tendrá un desarrollo en serie cosenoidal.

$$a_0 = \frac{1}{T}\int_{-\tau/2}^{\tau/2} U_{\max} dt = U_{maz}\frac{\tau}{T}$$

$$a_n = \frac{2}{T}\int_{-\tau/2}^{\tau/2} U_{\max}\cos\left(n\cdot\frac{2\pi}{T}\cdot t\right)dt = \frac{2U_{maz}}{\pi n} sen\left(n\cdot\pi\cdot\frac{\tau}{T}\right)$$

Para el caso específico de tener un periodo de 10 ms, un ancho del pulso de 5ms y que la amplitud sea de 5V, los coeficientes serán:

$$a_0 = 2.5V \quad a_1 = 3.18V(100Hz) \quad a_2 = 0V(200Hz) \quad a_3 = -1.06V(300Hz)$$
$$a_4 = 0V(400Hz) \quad a_5 = 0.635V(500Hz) \quad a_6 = 0V(600Hz) \quad a_7 = -0.45V(700Hz),...$$

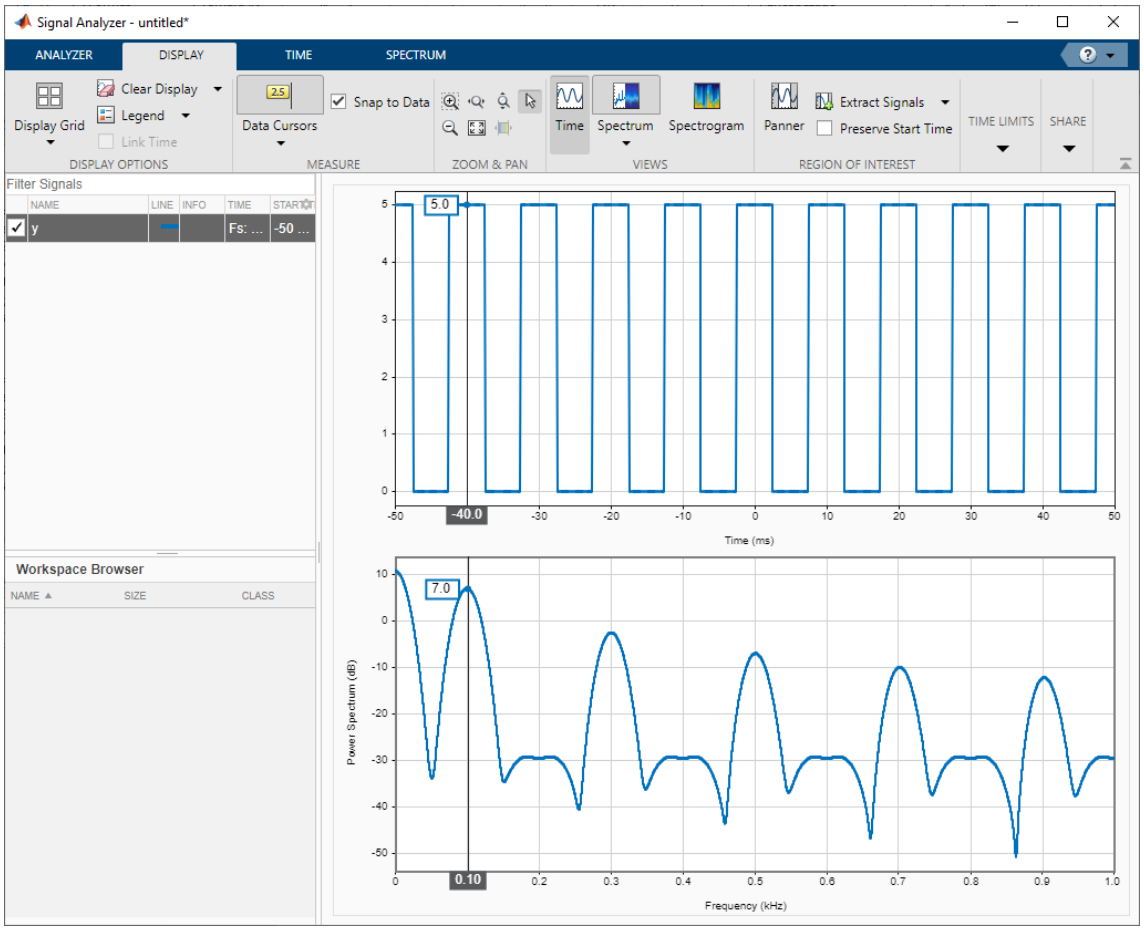


Con el comando **fft** se conseguirá obtener la transformada discreta de Fourier. Analizar y documentar la siguiente función de Matlab:

```
%% square
Ts=1e-4;
t=-50e-3:Ts:50e-3;
frec=100;
y=2.5*square(2*pi*frec*t+pi/2)+2.5;
figure(1);
plot(t,y);
axis([min(t),max(t),min(y)-0.2,max(y)+0.2]);

%% FFT
N_fft=2^8;
yw=fft(y,N_fft);
fs=1/Ts;
f=0:fs/(N_fft-1):fs;
figure(2);
plot(f,abs(yw)/N_fft);
figure(3);
plot(f,angle(yw));

%% Zoom
figure(4);
plot(f(f<1e3),abs(yw(f<1e3))/N_fft);
figure(5);
plot(f(f<1e3),angle(yw(f<1e3)));
```

Se recomienda que las variables, funciones y comentarios en los códigos estén en inglés.

También se puede determinar mediante la app de Matlab **Signal Analyzer**. Cargar los datos de la señal desde el workspace y aplicar el analizador de espectros. El análisis frecuencial es mostrado en la potencia espectral de la señal:

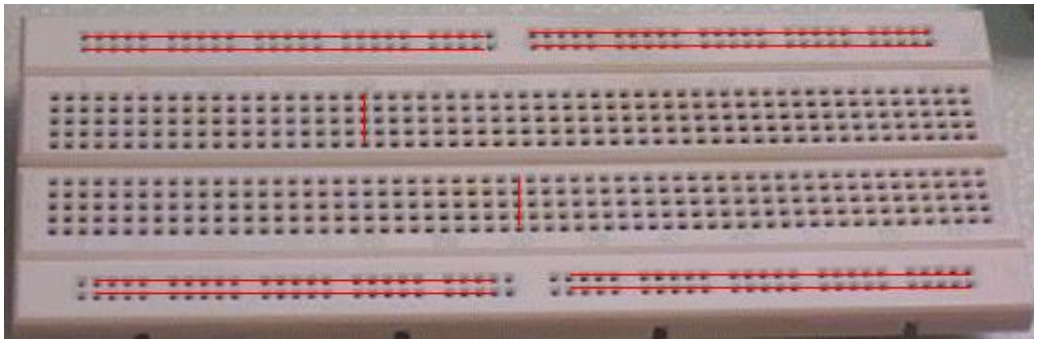


Tarea

Calcular el espectro de una señal senoidal y de dientes de sierra. Utilizar las funciones **sin()** y **sawtooth()**. Calcular la serie de Fourier de estas señales y explicar el sentido de cada tipo de espectro.

Mapa de conectividad de la protoboard

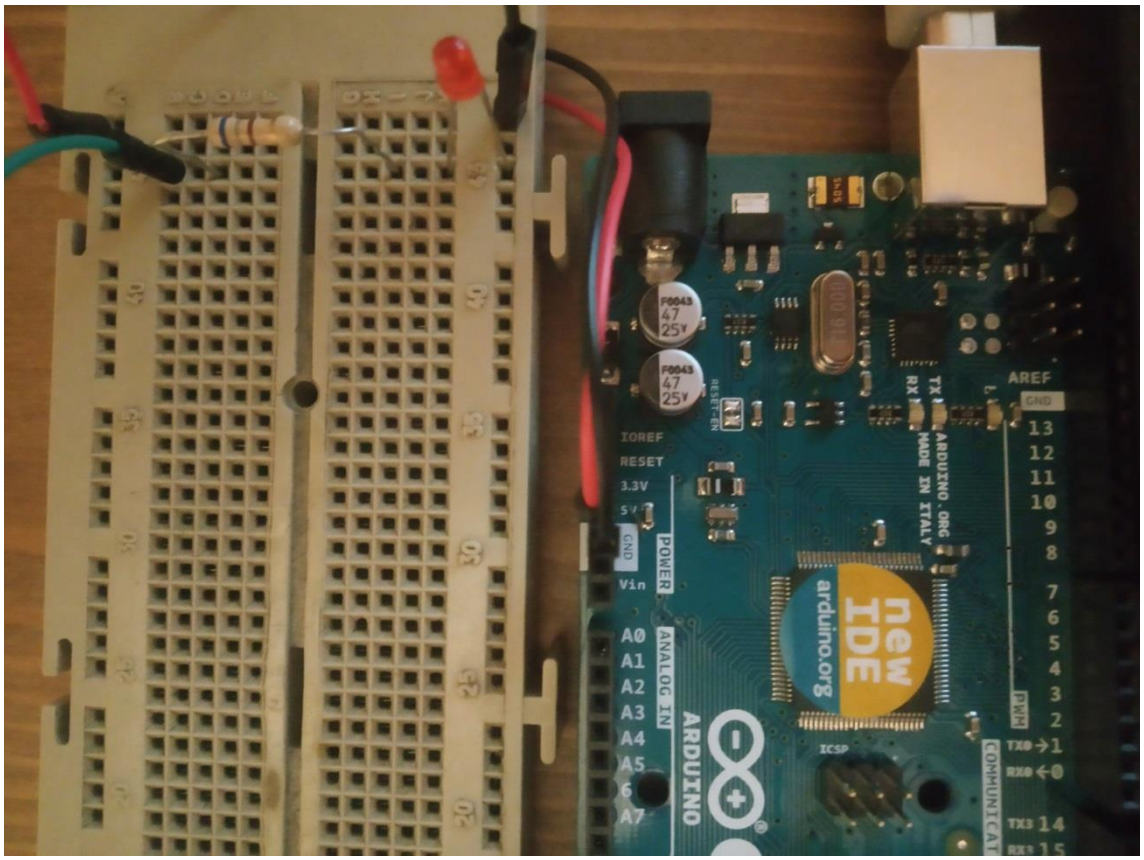
Desde la *protoboard* el alumno podrá realizar sus montajes mediante la inserción de los componentes y de los cables. La figura muestra en detalle este componente.

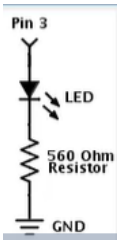


Verificar con el polímetro el mapa de conductividad de la protoboard

Control ON/OFF de un led mediante Arduino

La primera actividad con el Arduino consistirá el encendido y apagado de un led mediante una salida digital. Colocar un led con una resistencia limitadora de alrededor 680 ohmios. Verificar con los 5V de arduino que se enciende. Conectar el led con una salida digital del Arduino. La función writeDigitalPin será empleada para esta acción.





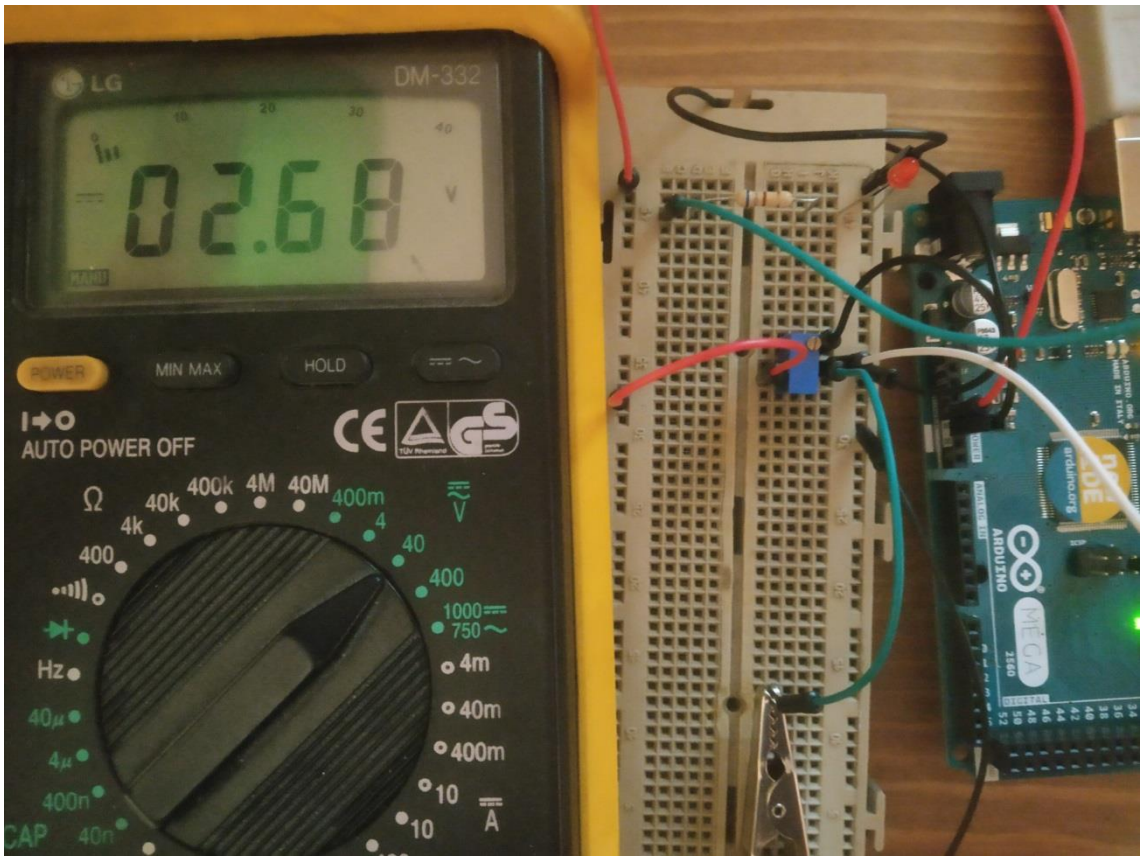

```
ard=arduino()
methods(ard)
port= 'D31';
writeDigitalPin(ard, port, 1);
writeDigitalPin(ard, port, 0);
%% Configure the LED to blink at a period of 0.5 second.
for i = 1:10
  writeDigitalPin(ard, port, 1);
  pause(0.5);
  writeDigitalPin(ard, port, 0);
  pause(0.5);
end
```

En la siguiente actividad se utilizará una salida digital con modulación ancho del pulso (*Pulse Width Modulation*, PWM). Habrá que cambiar la conexión a un puerto digital de salida que tenga PWM (ver mapas de entradas y salidas del Arduino). Observar los niveles de tensión en la salida PWM mediante el polímetro:

```
%% Digital PWM
port='D3';
brightness_step = (1-0)/20;
for i = 1:20
  writePWMDutyCycle(ard, port, i*brightness_step);
  pause(0.2);
end

for i = 1:20
  writePWMDutyCycle(ard, port, 1-i*brightness_step);
  pause(0.2);
end

%% Analog PWD
brightness_step = (5-0)/20;

for i = 1:20
  writePWMVoltage(ard, port, i*brightness_step);
  pause(0.2);
end

for i = 1:20
  writePWMVoltage(ard, port, 5-i*brightness_step);
  pause(0.1);
end
```

Lectura de una variable analógica

Para la lectura de una tensión se utiliza ***readVoltage***. Montar un potenciómetro de 100k alimentado entre 0V y 5V. El punto intermedio del potenciómetro conectar a una entrada analógica del Arduino. Verificar con el polímetro que el valor leído corresponde con el retorno de la función:




```
%% Checking analog port
port_Analog = 'A0';
readVoltage(ard, port_Analog)
```

A continuación, se procede a la regulación del nivel de excitación del led desde la señal de referencia marcada por el potenciómetro:

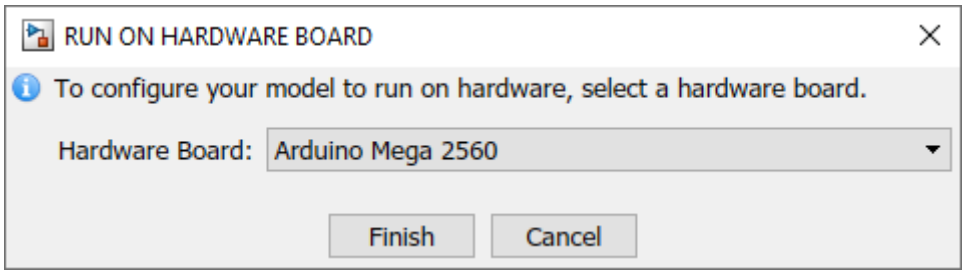
```
%% Control an LED using a potentiometer
port_Analog = 'A0';
time = 200;
while time > 0
  voltage = readVoltage(ard, port_Analog);
  writePWMVoltage(ard, port, voltage);

  time = time - 1;
  pause(0.1);
end
```

Al finalizar y antes de iniciar Simulink borrar todas las variables de Matlab, especialmente la variable **ard.** Utilizar:

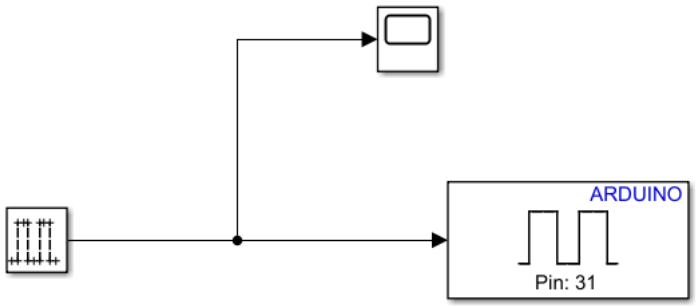
clear all

Control del Arduino desde Simulink

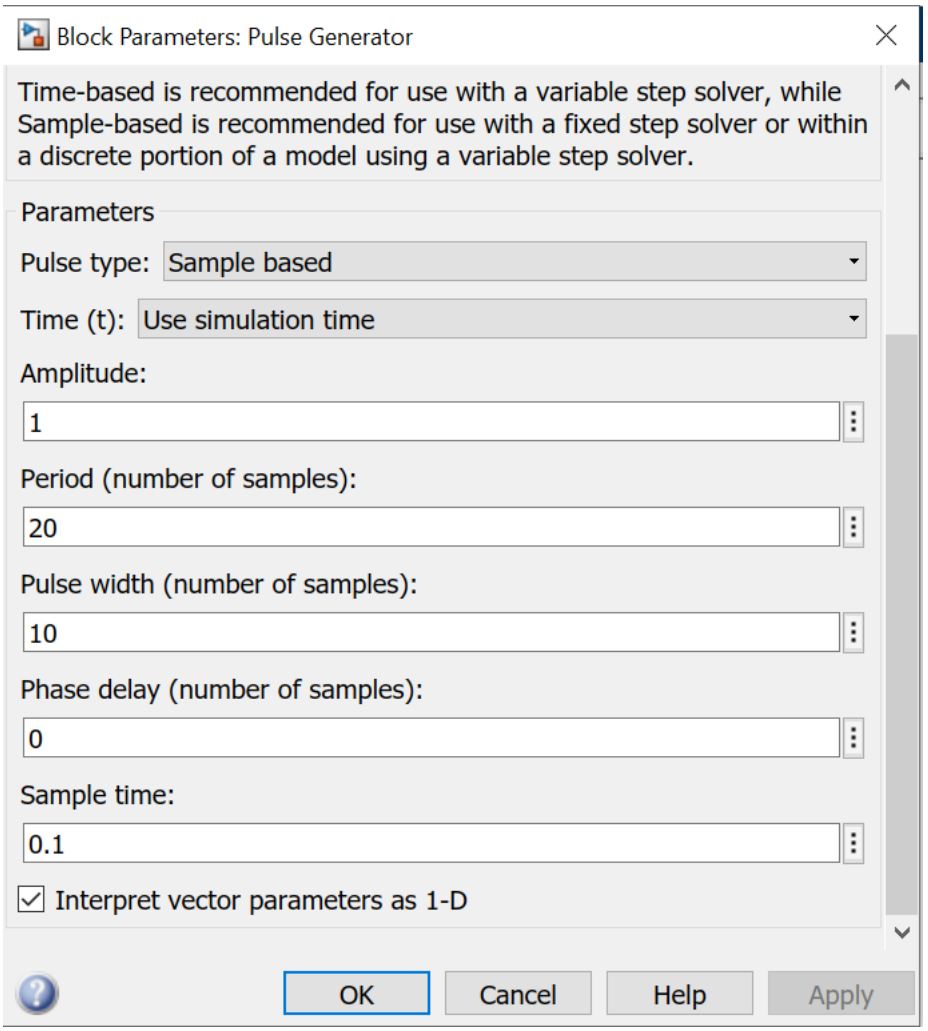
Los experimentos realizados desde Matlab con Arduino se van a trasladar a Simulink. En primer lugar, se procede al encendido ON/OFF del led. Activar la tarjeta de Arduino utilizando la App:



Observar que aparece la pestaña “Hardware”. Una vez seleccionado el hardware del Arduino, se procederá a construir el siguiente modelo:



Estará constituido por una salida digital, la visualización de la señal de excitación (Scope) y un generador de onda rectangular utilizando muestras cada 0.1 s. Las propiedades del generador quedan como:



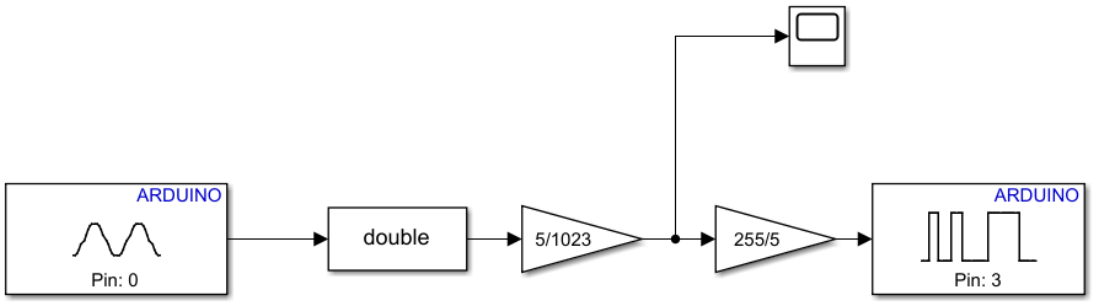
Tal cual queda definido, el led se activará durante 1s y se desactivará en otro segundo. Utilizar el tiempo de parada de forma infinita (inf), i.e. significa que el experimento estará activo hasta que el usuario lo pare. Opcionalmente se puede construir el modelo mediante *Buid Stand-Alone*. Ejecutar el modelo y observar tanto el led como scope. Una vez ejecutado, algunos modelos de Arduino, p.e. Mega 2560, permite la modificación de los parámetros del generador de pulso. Cambiar a un periodo de 4 segundos y con 1 s en estado ON.

Más información:

https://es.mathworks.com/videos/arduino-with-matlab-and-simulink-part-3-programming-arduino-with-simulink-99404.html

Control del led desde un potenciómetro

El nivel de tensión dado por el potenciómetro será el nivel de excitación en el led. Construir el siguiente modelo de simulink:



El convertidor analógico-digital del Arduino convierte la señal de 0-5V en un número natural de 0 a 1023. Esta señal se procesa convirtiéndola en punto flotante (double). Las salidas de PWM están ajustadas en el rango de 0-255. Por este motivo se realiza una adaptación de rangos con una ganancia k=255/1023.

Más información:

https://es.mathworks.com/videos/arduino-and-matlab-reading-inputs-and-writing-outputs-106502.html