

---

# TEST AUTOMATIZADOS DEL SISTEMA “GÜERTITO”

BASADO EN SIMULINK® REQUIREMENTS™ Y SIMULINK®  
TEST™

*Versión 1.1*

**GRUPO F:**

JORGE DEIRO FERRE (M19056)  
JOSE FIGUEROLA PALACIOS (M19080)  
JUAN RUBIO ROMERA (13399)  
SERGIO MARTÍNEZ HAMDOUN (M19155)

Ingeniería del Software  
Universidad Politécnica de Madrid  
10 de enero de 2021

# Contenido

---

<b>Introducción .....</b>	<b>1</b>
Propósito.....	1
Alcance de los test.....	1
<b>Realización de los requisitos en Simulink® Requirements™ .....</b>	<b>1</b>
Desarrollo.....	1
<b>Selección de los requisitos a evaluar mediante test automatizados .....</b>	<b>3</b>
Requisitos para testear .....	3
<b>Explicación de los test realizados (basados en los requisitos).....</b>	<b>6</b>
Introducción .....	6
Requisito cruzado con el sistema de puerta automática para gatos .....	6
RFU10 - Identificación única y personal en la aplicación .....	15
RFU04- Añadir huerto.....	25
RNFU03- Opción de control manual .....	28
RNFME01- Fallo en el riego .....	32
RNFD01 – Velocidad de actuación de los actuadores.....	35
RNFD02 – Recogida de información de los sensores .....	36
RNFME03 – Fallo en los sensores.....	39
RNFD04- Conexión con parte meteorológico .....	41
RNFD05- Permiso nivel usuario .....	44
<b>Conclusiones .....</b>	<b>49</b>
<b>Apéndices.....</b>	<b>49</b>

**Historial de Revisiones**

<b>Nombre</b>	<b>Fecha</b>	<b>Razón de los Cambios</b>	<b>Versión</b>
Versión inicial	06/01/2021	Redacción del documento y estructura general	1.0
Versión corregida	10/01/2021	Corrección y revisión del documento	1.1

# Introducción

---

## Propósito

El objetivo de este documento es doble. Por un lado, se pretende mostrar una parte de la implementación de nuestro sistema **Güertito**. Esta implementación se realizará utilizando Matlab y Simulink, tanto en forma de diagramas de estados como funciones definidas en Matlab.

El segundo objetivo deriva del primero, puesto que con estas pequeñas implementaciones podremos realizar, y como veremos a continuación, con éxito, diversos test automatizados para mostrar el correcto funcionamiento de las partes que se han implementado.

Cabe destacar que no se ha procedido a la implementación de un modelo completo del sistema. La razón es que, de esta manera, se consiguen implementar modelos más precisos y que se asemejen más a la realidad, para que una vez que se hayan desarrollado los test de forma exitosa, proceder a la implementación final en los microcontroladores.

Es importante resaltar que los test surgen para comprobar ciertos requisitos, y sobre estos requisitos se ha realizado la implementación de los modelos y los test automatizados. Por lo tanto, este documento está estructurado en dos partes: primero se muestran los requisitos programados con Simulink® Requirements™ de forma general, y posteriormente se escogen algunos requisitos para probar, con sus modelos correspondientes.

## Alcance de los test

Como ya se ha dicho, el objetivo de los test es el derivado de la propia definición de test. Además, es destacable que se han ido cambiando los test para que el resultado sea satisfactorio, por lo tanto, se mostrarán los resultados que han salido positivos. El nivel de detalle de los modelos deriva del cumplimiento de los requisitos que se han seleccionado para testear, siendo especialmente importante lo dicho por los requisitos para el desarrollo de los test y su resultado definitivo.

## Realización de los requisitos en Simulink® Requirements™

---

## Desarrollo

Para poder crear los requisitos se ha seguido un procedimiento similar al visto en la sesión de teoría de la asignatura Ingeniería del software. Se ha creado un proyecto de Matlab y en él, utilizando la

herramienta Simulink® Requirements™ se ha creado una tabla donde se han ido introduciendo todos los requisitos que ya se definieron en el documento de entra anterior llamado “*Especificación de Requisitos del Sistema Güertito*”. Se puede ver un ejemplo de la estructura en la *Ilustración 1*, y un ejemplo de requisito desarrollado en la *Ilustración 2*:











Index	ID	Summary
▼  Requisitos_Güertito*		
▼  1	RC	REQUISITOS CRUZADOS
 1.1	#7	Cerradura
 1.2	#8	Cocina
 1.3	#9	Puerta para gatos
▼  2	RF	FUNCIONALES
>  2.1	RFS	Requisitos de sistema
>  2.2	RU	Requisitos de Usuario
▼  3	RNF	REQUISITOS NO FUNCIONALES
>  3.1	RNFD	Requisitos Desempeño
>  3.2	RNFS	Requisitos Seguridad
>  3.3	RNFME	Requisitos de Manejo de Errores
>  3.4	RNFU	Requisitos de Usabilidad
>  3.5	RNFD	Requisitos de Disponibilidad
>  3.6	RNFN	Requisitos de Normativa

Ilustración 1: Estructura de los requisitos del sistema Güertito

The screenshot shows a requirements management interface. On the left, a tree view displays a hierarchy of requirements under 'Requisitos\_Güertito\*'. The tree includes categories like 'REQUISITOS CRUZADOS', 'FUNCIONALES', and 'REQUISITOS NO FUNCIONALES'. Requirement '2.1.2 RFS02 Regar huerto' is selected. On the right, the 'Properties' panel for RFS02 is shown. It includes fields for 'Type' (Functional), 'Index' (2.1.2), 'Custom ID' (RFS02), and 'Summary' (Regar huerto). The 'Description' tab is active, showing a detailed description of the irrigation system's capabilities and user options. Below the description, there are fields for 'Keywords', 'Revision information', 'Links', and 'Comments'.

Index	ID	Summary
1	RC	REQUISITOS CRUZADOS
1.1	#7	Cerradura
1.2	#8	Cocina
1.3	#9	Puerta para gatos
2	RF	FUNCIONALES
2.1	RFS	Requisitos de sistema
2.1.1	RFS01	Monitorizar parámetros de cultivo
2.1.2	RFS02	Regar huerto
2.1.3	RFS03	Acceder a datos meteorológicos
2.1.4	RFS04	Comunicación con otros sistemas
2.1.5	RFS05	Estimación de recogida de los al...
2.1.6	RFS06	Acceso a base de datos de difer...
2.2	RU	Requisitos de Usuario
3	RNF	REQUISITOS NO FUNCIONALES
3.1	RNFD	Requisitos Desempeño
3.2	RNFS	Requisitos Seguridad
3.3	RNFME	Requisitos de Manejo de Errores
3.4	RNFU	Requisitos de Usabilidad
3.5	RNFD	Requisitos de Disponibilidad
3.6	RNFN	Requisitos de Normativa

**Properties**

Type: Functional

Index: 2.1.2

Custom ID: RFS02

Summary: Regar huerto

**Description**

El sistema debe ser capaz de regar el huerto a través de los actuadores integrados en él de manera adecuada para la salud de la planta a partir de los parámetros determinantes (temperatura, humedad, viento, calidad del aire, estado de la tierra, estado de la planta, fecha) extraídos de los sensores, la información meteorológica y la propia aplicación. Esta actuación se realizará por el propio usuario y ofrecerá diversas opciones:

- Regar el huerto de manera inmediata: el usuario, activando esta opción, provoca que el huerto comience a regarse de forma inmediata.
- Programar riego del huerto: esta opción permite que se active el riego del huerto de forma remota, mediante la programación de la hora de inicio y la duración del riego.

Cabe destacar que la intensidad de riego del huerto será la misma en ambos casos.

**Keywords:**

**Revision information:**

**Links**

No links

**Comments**

Ilustración 2: Ejemplo de requisito implementado del sistema Güertito

Una vez que se han implementado los requisitos, se ha procedido a seleccionar aquellos a evaluar, con la implementación de los modelos correspondientes.

## Selección de los requisitos a evaluar mediante test automatizados

### Requisitos para testear

Tabla 1: Tabla con los requisitos probados

CÓDIGO	TÍTULO	REQUISITO
<b>Requisito cruzado</b>	Puerta para gatos	La aplicación de la gatera se conectará por una API a los datos recopilados por el sistema del huerto acerca de la meteorología, y avisará en caso de precipitaciones para

		poder programar acciones como cerrar la gatera para evitar que el gato salga de la vivienda.
<b>RFU10</b>	Identificación única y personal en la aplicación	El sistema debe proporcionar un usuario único y personal a cada persona que utilice la aplicación, debiéndose verificar la verdadera identidad de la persona que está usando un determinado usuario. El login en la aplicación se realizará mediante un sistema de verificación en dos pasos, estableciendo una clave única para cada usuario y una verificación vía SMS o correo electrónico. Como precondition se establece que el usuario esté dado de alta en el sistema y haya realizado dicha verificación. Una vez dado de alta se le asignar se podrá asignar un rol al usuario tal y como se indica en el requisito RFU11.
<b>RFU04</b>	Añadir huerto	El usuario debe ser capaz de añadir huertos al sistema. Para ello, como precondition se establece que el usuario esté dado de alta en el sistema, además de haber realizado el log in con anterioridad. Una vez realizada la función, el nuevo huerto aparecerá en la interfaz con el nombre especificado por el usuario, precondition básica para los requisitos RFU05, RFU06 y RFU07.
<b>RNFU03</b>	Opción de control manual	Se debe poder de controlar manualmente todas aquellas características que sean planificadas para ser ejecutadas de forma automática por el sistema. En caso de que la comunicación entre las diferentes partes del sistema falle, así como nuestro servidor, los usuarios dispondrán del control manual de los dispositivos para que este siga funcionando correctamente hasta que se solucionen los problemas aparecidos.
<b>RNFME01</b>	Fallo en el riego	El sistema debe notificar y reportar al usuario en caso de fallo en el riego vía la aplicación para la manipulación manual posterior. En este tipo de fallo se incluyen, el fallo de un actuador, así como los requisitos RNFD01 Y RNFD03. El aviso de fallo se debe dar como máximo en los 5 segundos posteriores al fallo del actuador, para corroborar que no es un cambio de decisión en el usuario.
<b>RNFME03</b>	Fallo en los sensores	El sistema debe notificar y reportar al usuario en caso de fallo en los sensores vía la aplicación para la reparación posterior. Esto se detectará por medio del requisito RNFD02. En caso de que se produzca este tipo de error, se permitirá realizar las acciones típicas del sistema al usuario, tales como activar el riego, abonos, luces, humedad, pero dejando bien claro que queda bajo su decisión debido a que los sensores no son capaces de recabar datos correctamente.
<b>RNFD01</b>	Velocidad de actuación de los actuadores	Los actuadores deben responder antes de 1 segundo tras mandar una acción de actuación desde la aplicación. Esto ocurrirá cuando el sistema lo considere oportuno debido a condiciones meteorológicas o condiciones específicas de la planta extraídas directamente de las bases de datos. Además, como ya se dijo en el requisito RFS02, también se debe cumplir esa velocidad para cuando el usuario selecciona la opción de regar desde la app. Sin embargo, este tiempo de respuesta de 1 segundo se puede aumentar hasta los 3 segundos cuando la función de

		regar se produce de forma autónoma, es decir, mediante la programación de un riego.
<b>RNFD02</b>	Recogida de información de los sensores	Los sensores deberán recoger información en todo momento que sea requerido por el sistema y enviar la información correctamente de vuelta al sistema para que este evalúe la situación. En concreto, los sensores de temperatura y humedad deben enviar información a la placa cada 2 segundos, teniendo la capacidad de transmitir la señal de hasta 20 metros. El sensor de luminosidad, encargado de enviar los datos a los actuadores de luz, suelen tener un tiempo de respuesta de entre 25 y 30 ms. Esta acción será requerida por el sistema al menos una vez cada tres días para evaluar la evolución del huerto, en caso de que el sistema detecte algún problema en la evolución este pedirá información 1 vez al día a lo largo de la semana siguiente a la realización de los cambios efectuados para solventar el problema en el huerto.
<b>RNFD04</b>	Conexión con parte meteorológico	El sistema deberá comunicarse con el parte meteorológico una vez al día. Esto ocurrirá independientemente de lo que requiera el usuario, el que podrá pedir al sistema que actualice más veces durante el día en caso de que él lo requiera. En cada actualización recogerá la información de los próximos 2 días acerca de humedad, temperatura, precipitaciones, viento y calidad del aire. Además, se debe ir actualizando el parte meteorológico cada media hora, tal y como se ha especificado en el requisito RFS03. Se debe verificar, además, a través de un mensaje mostrado en la app, que el parte se ha actualizado correctamente. En caso de no conectarse al parte meteorológico, se mostrará un aviso de error en la aplicación, para que el usuario compruebe su conexión a internet. En cualquier caso, el sistema deberá dejar al usuario acceder al resto de funcionalidades.
<b>RNFD05</b>	Permisos nivel usuario	El sistema, tras el registro del usuario, en función de la participación sobre el huerto en concreto que tendrá se le asignará un nivel de usuario que posteriormente le servirá para poder actuar sobre las distintas funcionalidades del sistema. El usuario podrá acceder a estas funcionalidades siempre y cuando se encuentre igual o por debajo de su nivel de usuario, de esta manera se controlarán los cambios de mejor manera y al gusto del creador del huerto.

\*Los códigos de los requisitos vienen explicados al final del documento, en el apéndice A: Glosario

Para cada requisito se ha desarrollado un modelo de implementación. Además, se ha establecido una relación entre los requisitos probados y los test implementados.

Tabla 2: Relación entre los requisitos y los números de test aplicados a cada uno de ellos

<b>REQUISITO</b>	<b>NÚMERO DE TEST</b>
<i>Requisito cruzado</i>	3
<i>RFU10</i>	3
<i>RFU04</i>	1



<i>RNFU03</i>	2
<i>RNFME01</i>	2
<i>RNFME03</i>	2
<i>RNFD01</i>	1
<i>RNFD02</i>	1
<i>RNFD04</i>	2
<i>RNFD05</i>	2

## Explicación de los test realizados (basados en los requisitos)

### Introducción

Se presentan, por lo tanto, los modelos de implementación de cada requisito, con la explicación de los test que se han llevado a cabo sobre cada modelo. El hecho de incluir varios test dentro de cada modelo se basa en mostrar el comportamiento distinto del sistema ante un cambio de entradas. Todo esto se mostrará por el cambio de las señales de salida.

### Requisito cruzado con el sistema de puerta automática para gatos

El modelo de implementación que conecte nuestro sistema con el sistema de apertura y cierre automático para gatos se basa en que, cuando se detecten precipitaciones cercanas, se genere una señal de alarma que se envíe a dicho sistema. Por lo tanto, lo que se ha implementado es una secuencia mediante la cual se accedan a los datos meteorológicos. Si se ha accedido de forma correcta a la información meteorológica, se genera un pulso de salida en la señal "*pronóstico*". El siguiente paso consiste en acceder a la información de precipitaciones, sólo si se ha accedido de forma correcta al pronóstico temporal. Si se obtiene la información de precipitaciones, se pasa a evaluar si son cercanas o no, parámetro que se determinará por la señal "*lluvia*", que sigue la siguiente lógica:

*if (lluvia==1)*

*Alarma*

*if (lluvia == 2)*

*No\_alarma*

La señal de alarma o no alarma viene definida por “*mensaje\_gatera*”, que envía esta información codificada al sistema de la puerta automática para gatos, para que posteriormente este sistema actúe en consecuencia de esta información enviada.

Se ha implementado un sistema de fallos, de tal forma que, si no hay conexión a internet, se incrementen el número de intentos que de forma automática el sistema acceda a internet para recopilar los datos meteorológicos, y así se mande un mensaje de error a la aplicación del usuario del sistema Güertito, para que lo vuelva a intentar. Esta señal viene implementada como “error”.

Por lo tanto, el sistema modelado derivado de este requisito tiene las siguientes señales de entrada y de salida:

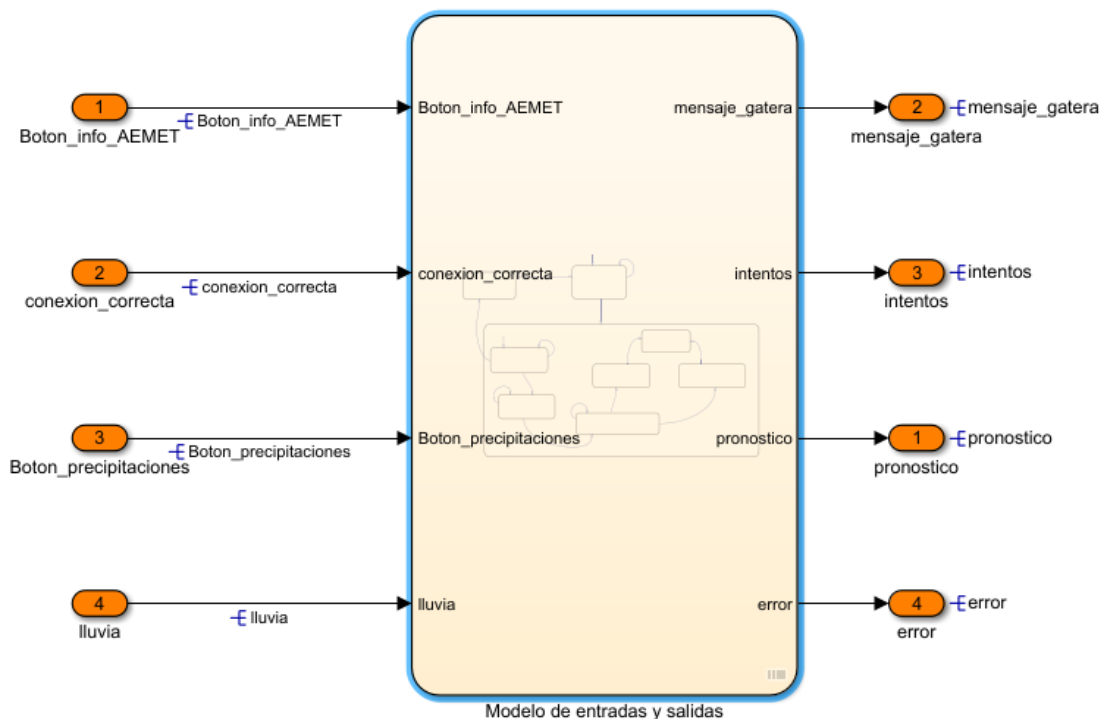


Ilustración 3: Modelo de entradas y salidas

Se ha intentado modelar de alguna forma los botones que el usuario tiene que pulsar para acceder a la información de las precipitaciones, que son “*Boton\_info\_AEMET*” y “*Boton\_precipitaciones*”. Además, se ha creado la señal de “*intentos*” simplemente con el objetivo de monitorizar los intentos que el sistema realiza para acceder de forma correcta a la información del tiempo. Por lo tanto, el modelo que resulta, teniendo en cuenta todo lo anterior, es el siguiente:

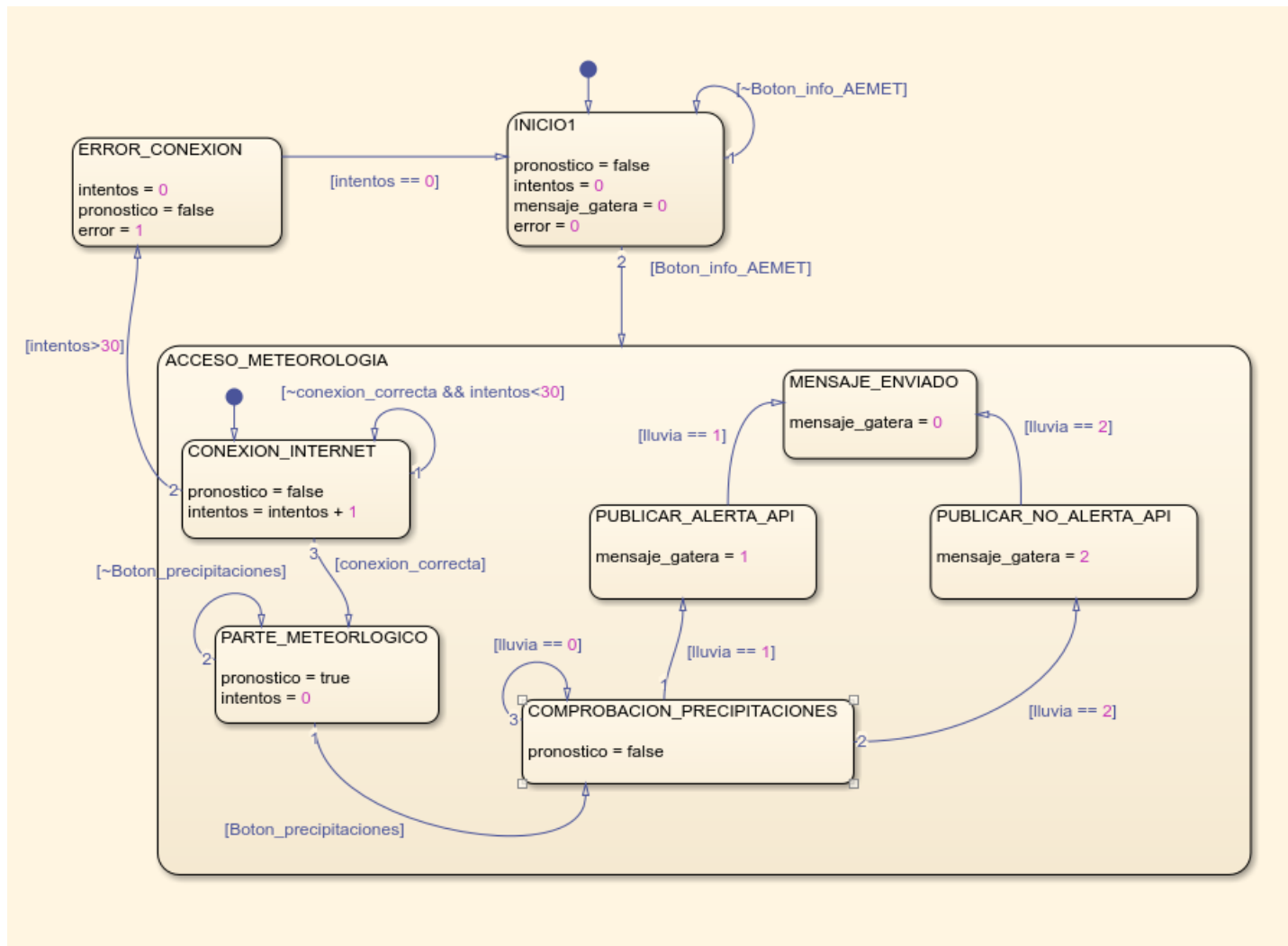


Ilustración 4: Modelo de conexión con el sistema de la puerta de gatos que genera una alerta de lluvias

Sobre este modelo se han realizado los test que se detallan a continuación:

1. **Test1:** Comprobación de se accede correctamente al pronóstico del tiempo, se accede a las precipitaciones, pero estas son lo suficientemente lejanas como para no mandar una señal de alarma al sistema de la puerta de la gatera. El flujograma del test realizado es el siguiente, mostrándose los cambios que se desean en las diversas señales de entrada, así como las transiciones con sus descripciones:

Step	Transition	Next Step	Description
<b>: Run</b>  <pre>%% Initialize data outputs. Boton_info_AEMET = false; conexion_correcta = false; Boton_precipitaciones = false; lluvia = 0;</pre> <small>Add step after • Add sub-step • ⚙</small>	1. <code>after(1,msec)</code>	step_1 ▼	Inicialización de los parámetros. Se coloca el valor de la lluvia a cero para que posteriormente se modifique con lo que se reciba de la información del tiempo
<b>step_1</b>  <pre>Boton_info_AEMET = true;</pre>	1. <code>after(2,sec)</code>	step_2 ▼	Se pulsa el botón de acceso a los datos de la AEMET
<b>step_2</b>  <pre>conexion_correcta = true; Boton_precipitaciones = true;</pre>	1. <code>after(1,msec)</code>	step_3 ▼	La conexión a internet es correcta y se pulsa el botón de precipitaciones
<b>step_3</b>  <pre>lluvia = 2;</pre>			No se detectan lluvias cercanas por lo que el valor de lluvia toma el valor de '2'. Se produce la salida correspondiente en la variable 'mensaje_gatera'

Ilustración 5: Flujograma del test1

En cuanto a la salida, vemos como efectivamente se recibe el pronóstico del tiempo correctamente (**señal azul**) y la señal de lluvia toma el valor de 2 (**señal naranja**), indicando que no hay lluvias cercanas:

## Simulation Data Inspector: Inspect

Report Generated 07-Jan-2021 16:12:37

### Run 1: Comm\_gatera\_TEST\_a

Name	Line
Output Conversion Subsystem:1	—
Output Conversion Subsystem:2	—

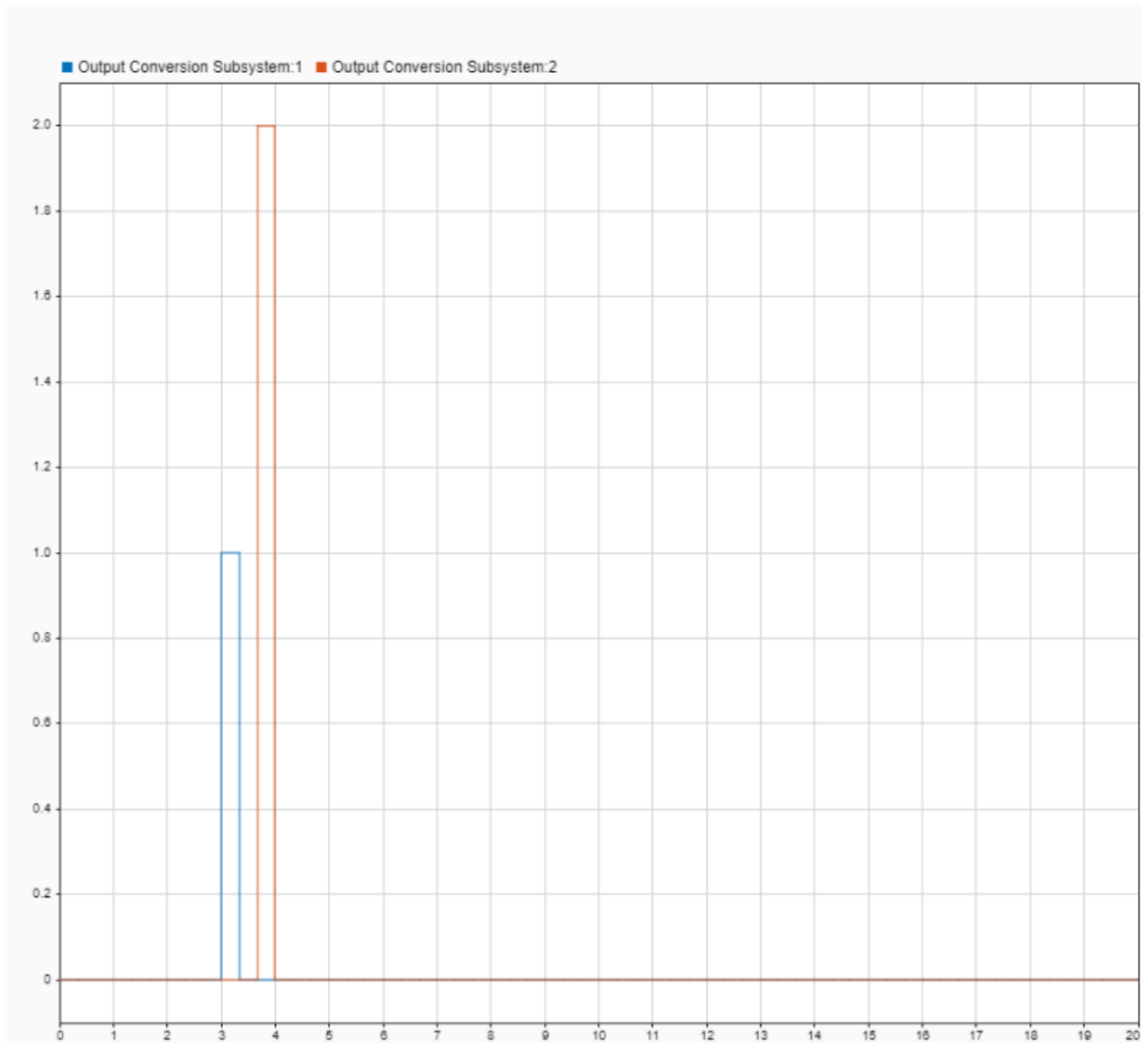


Ilustración 6: Resultado en las señales de salida del Test1

- Test2:** se sigue el mismo procedimiento que el caso anterior, pero esta vez sí que hay precipitaciones cercanas (**señal naranja**), por lo tanto la señal de lluvia adoptará un valor de

1, y mandando la señal de alarma o aviso al sistema de la puerta para gatos. También se comprueba que se recibe de forma satisfactoria el pronóstico (señal azul). El código de implementación es el siguiente:

Step	Transition	Next Step	Description
<b>Run</b>  <pre>%% Initialize data outputs. Boton_info_AEMET = false; conexion_correcta = false; Boton_precipitaciones = false; Lluvia = 0;</pre>	1. <code>after(1,msec)</code>	step_1 ▼	Inicialización de los parámetros. Se coloca el valor de la lluvia a cero para que posteriormente se modifique con lo que se reciba de la información del tiempo
<b>step_1</b>  <pre>Boton_info_AEMET = true;</pre>	1. <code>after(2,sec)</code>	step_2 ▼	Se pulsa el botón de acceso a los datos de la AEMET
<b>step_2</b>  <pre>conexion_correcta = true; Boton_precipitaciones = true;</pre>	1. <code>after(1,msec)</code>	step_3 ▼	La conexión a internet es correcta y se pulsa el botón de precipitaciones
<b>step_3</b>  <pre>Lluvia = 1;</pre>			Se detectan lluvias cercanas por lo que el valor de lluvia toma el valor de '1'. Se produce la salida correspondiente en la variable 'mensaje_gatera'

Ilustración 7: Implementación del código para el Test2

Y por lo tanto, en la salida tendremos un '1' como respuesta al valor de lluvias próximas:

## Simulation Data Inspector: Inspect

Report Generated 07-Jan-2021 16:18:36

### Run 2: Comm\_gatera\_TEST\_b

Name	Line
Output Conversion Subsystem:1	—
Output Conversion Subsystem:2	—

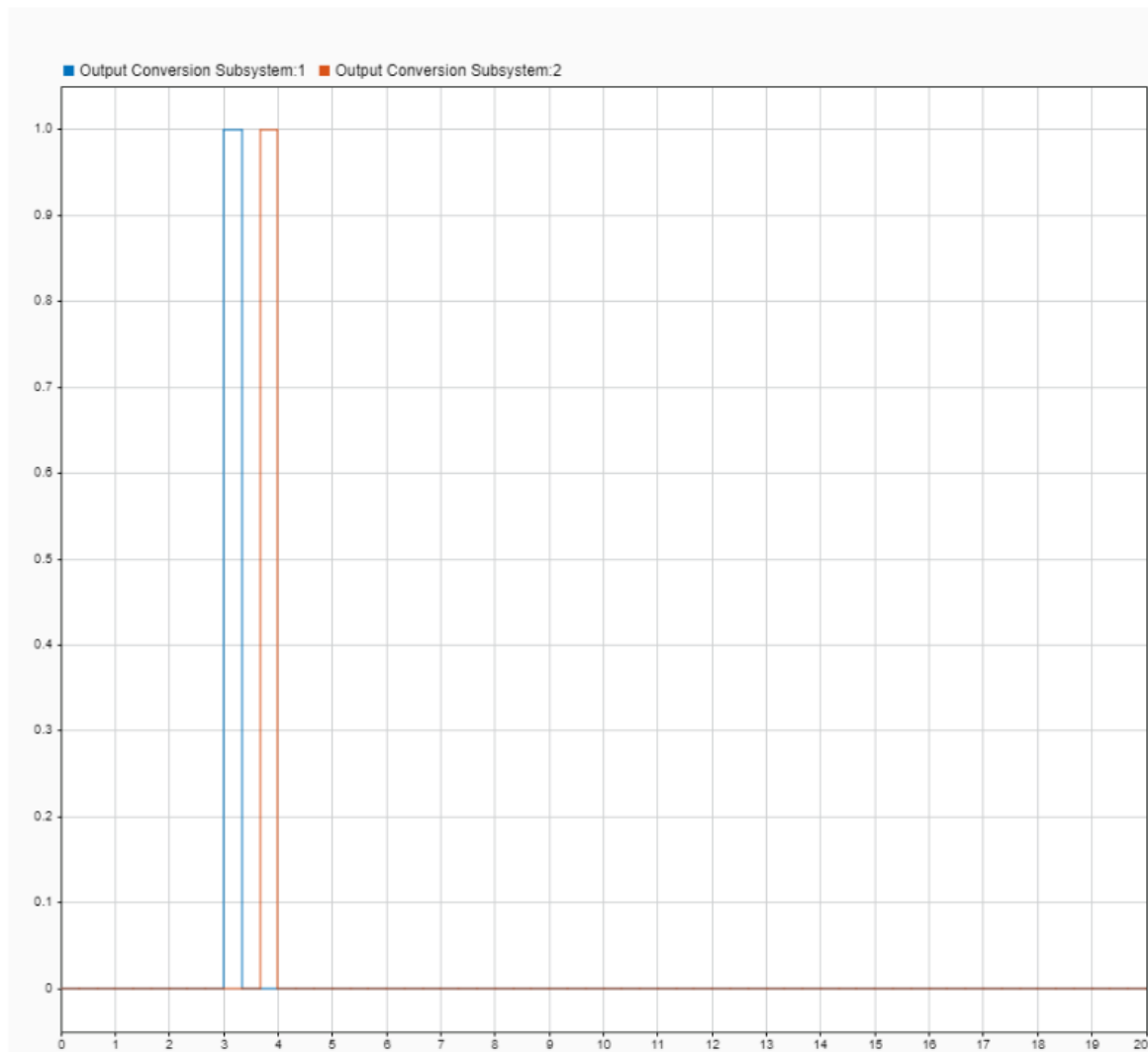


Ilustración 8: Resultado en las señales de salida del Test2

- Test3:** Este test es ligeramente diferente a los anteriores, porque ahora lo que se está testeando es que no hay conexión a internet, por lo que la señal de pronóstico será 0 y cuando se alcance un número de intentos determinados, pasará al estado de error (**señal azul**), se reinician los intentos y se vuelve al inicio de la aplicación, en la que el usuario tenga que volver a darle al botón de obtención de datos meteorológicos.

Step	Transition	Next Step	Description
<b>Run</b>  %% Initialize data outputs. Boton_info_AEMET = false; Boton_precipitaciones = false; lluvia = 0;	1. after(1,msec)	step_1 ▼	Inicialización de los parámetros. Se coloca el valor de la lluvia a cero para que posteriormente se modifique con lo que se reciba de la información del tiempo
<b>step_1</b>  Boton_info_AEMET = true;	1. after(1,msec)	step_2 ▼	Se pulsa el botón de acceso a los datos de la AEMET
<b>step_2</b>  conexion_correcta = false;			Como la conexión no es correcta, se sumarán los intentos hasta que se sobrepasen y se activará la señal de error

Ilustración 9: Implementación del Test3

Y el resultado del test es el siguiente:



## Simulation Data Inspector: Inspect

Report Generated 07-Jan-2021 16:25:59

### Run 3: Comm\_gatera\_TEST\_c

Name	Line
Output Conversion Subsystem:4	—

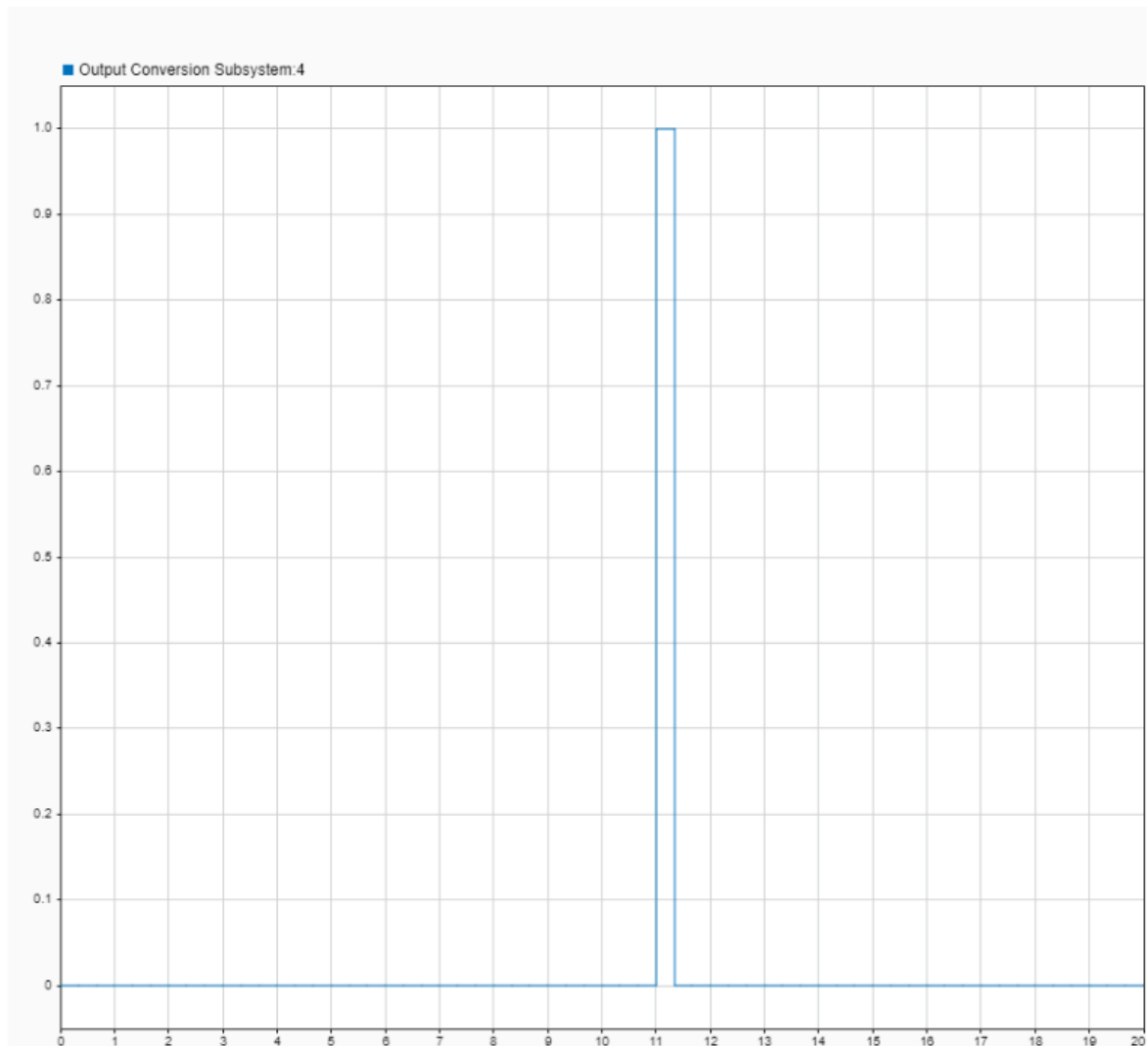


Ilustración 10: Resultados de la señal de salida del Test3

A modo de curiosidad, se muestra la señal de la variable **intentos** (señal azul), que se ha puesto como salida para vislumbrar que, efectivamente, cuando se sobrepasan los intentos (señal naranja) de acceso al tiempo, se activa la señal de error:

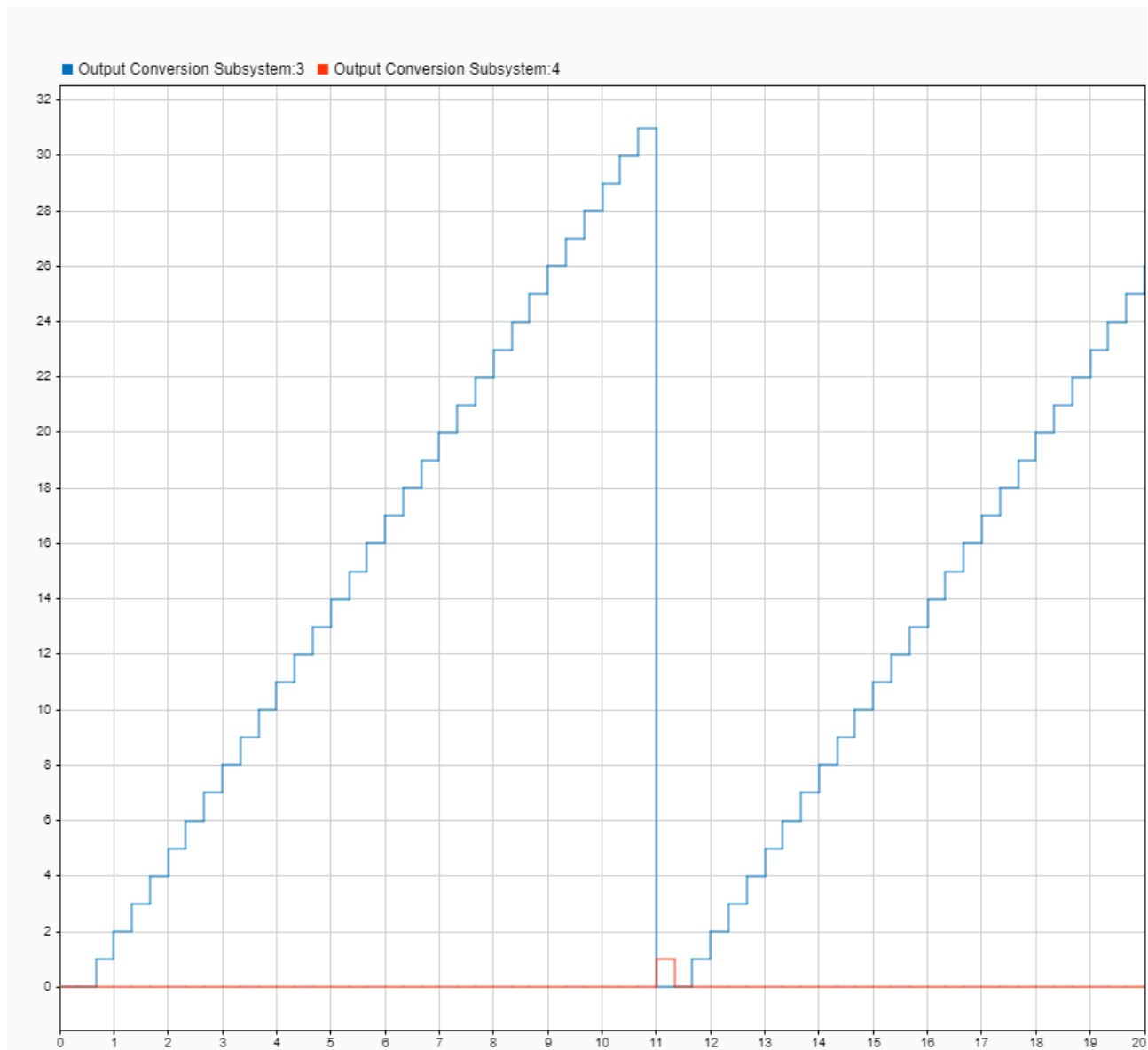


Ilustración 11: resultados de la salida cuando se realiza el Test3

Con esto ya quedaría verificado el requisito cruzado de comunicación con la gatera. Podría haber una mejora a este requisito y a su correspondiente verificación, que sería aumentar el número de casos del tiempo. Por ejemplo, se podría comprobar que, en vez de lluvia, sea un pronóstico de nieve, de vientos fuertes, de temperaturas extremadamente bajas... con el objetivo de que se cierre la puerta de la gatera y el gato no sufra condiciones adversas.

## RFU10 - Identificación única y personal en la aplicación

El modelo de implementación que permita al usuario registrarse y posteriormente realizar el login es algo más complejo que el anterior, pues hay una gran diversidad de casos. En esta primera

implementación se han tenido casi todos en cuenta, si bien es cierto que se podría completar con algunos más en versiones posteriores.

Este sistema simplemente detecta lo que desea el futuro usuario de la aplicación. Mediante la variable '*Selector*', se recoge ese deseo, de bien registrarse o logearse.

En caso de que el usuario decida **registrarse** (Selector tiene valor de 1) entonces el usuario pasará por una serie de etapas, representadas en forma de un diagrama de estados, y no se podrá pasar a la siguiente etapa hasta que no complete todas las fases. Se pasará por un estado de introducir correo, otro de introducir contraseña y otro de introducir otros datos complementarios y secundarios.

Después de esto, se pasará a la fase de **login**, de forma obligada. En esta primera fase de registro, se deberá verificar mediante una verificación en dos pasos el nuevo usuario. En el login se siguen los mismos pasos pero sin el estado de introducir esa información secundaria considerada en el registro: estado de introducir correo, introducir contraseña y verificación en dos pasos de ese usuario. Hay que destacar que la verificación en dos pasos se realizará mediante un envío de un correo electrónico al introducido por el usuario momentos antes. Cuando el usuario pasa a la fase de login, la variable Selector cambia de valor, de tal forma que ya no podrá volver a la fase de registro. En este momento la variable '*Selector*' valdrá 2.

Además, en afán de hacer de forma más eficiente la implementación, se han utilizado las mismas variables intermedias de los estados de introducir correo, introducir contraseña y verificación en dos pasos, las cuales se reinician a *false* en caso de haber superado el login correctamente.

Por último, en cuanto el usuario ha pasado las fases de registro y login de forma satisfactoria, se vuelve al estado inicial del menú de la app.

Se han considerado una serie de posibles fallos, bien sea debido a la conexión con el servidor, o bien porque el usuario no ha seguido las indicaciones de seguridad a la hora de crear la contraseña. Esto se ha representado, al igual que el requisito anterior, mediante dos variables de '*error*' e '*intentos*'.

Las entradas utilizadas y las salidas monitorizadas son las siguientes:

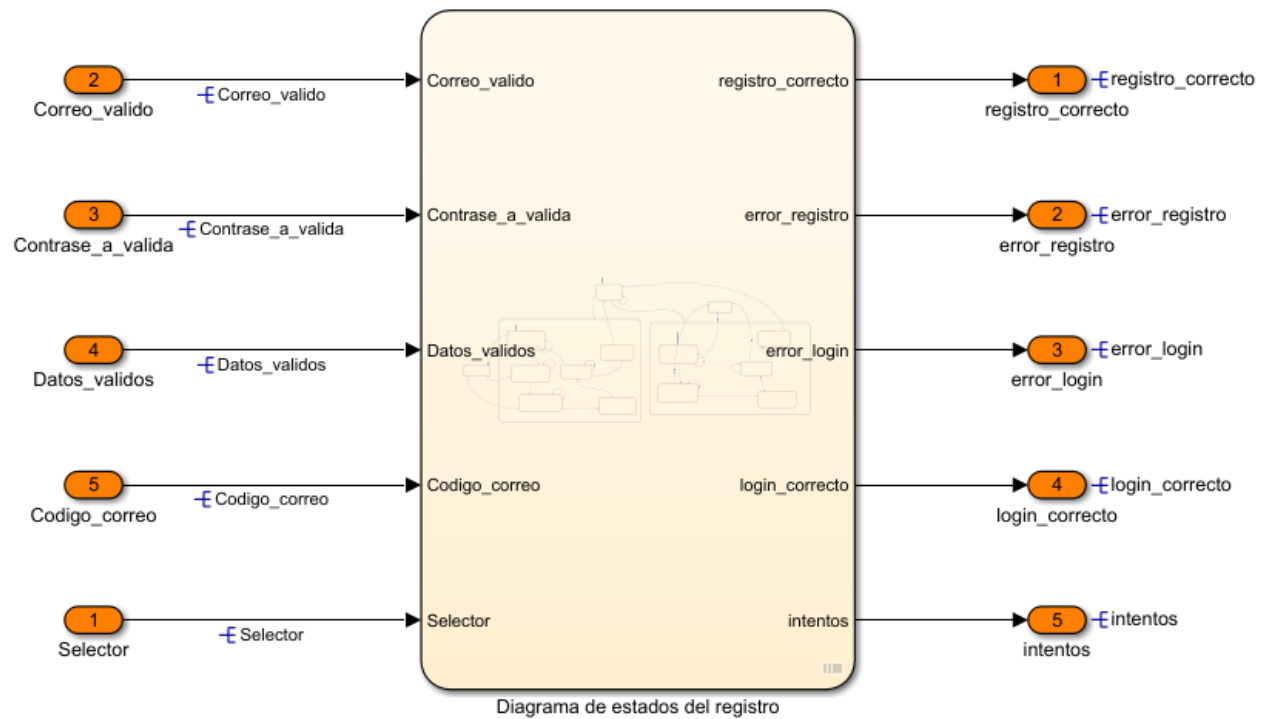


Ilustración 12: entradas y salidas del registro y login

Se ha identificado dos tipos de error bien sea durante la fase de registro o durante la fase de login, por los motivos comentados más arriba. El diagrama de estados queda de la siguiente manera:

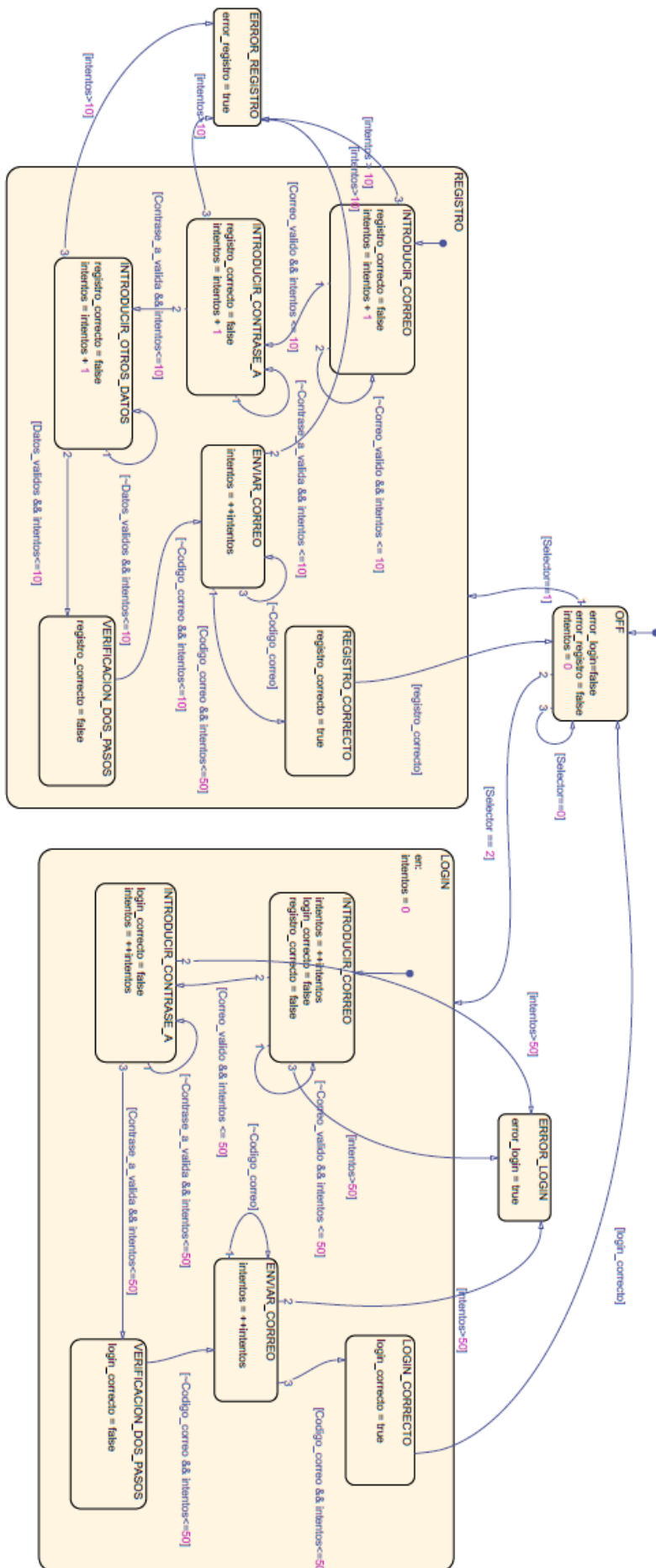


Ilustración 13: Diagrama de estados del registro y login

Sobre este modelo se ha procedido a implementar una serie de test. En concreto, se han implementado 3 test que tienen que ver con el caso de que un usuario se registre y realice el login correctamente, un usuario que solamente realice el login, y un caso de error en el registro.

- **Test1:** con la variable '*Selector*' a 0, se procede a registrarse un usuario. Se cambia el valor de la variable '*Selector*', poniéndose a 1, y poco a poco comienza esa secuencia en la cual el usuario se registra:
  - Introducir correo
  - Introducir contraseña
  - Introducir otros datos
  - Verificación en dos pasos
  - Enviar correo y confirmar
  - Registro correcto

Cuando el usuario se ha registrado correctamente, es importante que las señales anteriores que se han ido poniendo a *true*, cambien de estado a *false*, porque a continuación se pasará al login. El login es el mismo proceso descrito antes, solo que aquí no hay una introducción de datos secundarios. Cuando se ha producido el login correcto, entonces la señal *Selector* cambia a 0, porque ya no es necesario que comience ninguna otra secuencia.

El código de este test es el siguiente:

Step	Transition	Next Step
<b>Run</b>  %% Initialize data outputs. Selector = 0; Correo_valido = <b>false</b> ; Contrase_a_valida = <b>false</b> ; Datos_validos = <b>false</b> ; Codigo_correo = <b>false</b> ; 	1. <b>after</b> (1,msec)	step_1 ▼
<b>step_1</b>  Selector = 1; 	1. <b>after</b> (1,sec)	step_2 ▼
<b>step_2</b>  Correo_valido = <b>true</b> ; 	1. <b>after</b> (1,sec)	step_3 ▼
<b>step_3</b>  Contrase_a_valida = <b>true</b> ; 	1. <b>after</b> (1,sec)	step_4 ▼
<b>step_4</b>  Datos_validos = <b>true</b> ; 	1. <b>after</b> (1,sec)	step_5 ▼
<b>step_5</b>  Codigo_correo = <b>true</b> ; Selector = 2; 	1. <b>after</b> (1,sec)	step_6 ▼
<b>step_6</b>  Correo_valido = <b>false</b> ; Contrase_a_valida = <b>false</b> ; Datos_validos = <b>false</b> ; Codigo_correo = <b>false</b> ;  <b>if</b> (login_correcto) Selector = 0 <b>end</b> 	1. <b>after</b> (1,sec) 2. Selector == 0	step_2 ▼ step_7 ▼
<b>step_7</b>  Selector = 0; 		

Ilustración 14: Código del test1

Y como resultado se sacarán las señales de *registro\_correcto* (señal azul), y *login\_correcto* (señal naranja):

## Simulation Data Inspector: Inspect

Report Generated 07-Jan-2021 17:28:58

### Run 6: Registro\_TEST\_a

Name	Line
Output Conversion Subsystem:1	—
Output Conversion Subsystem:4	—

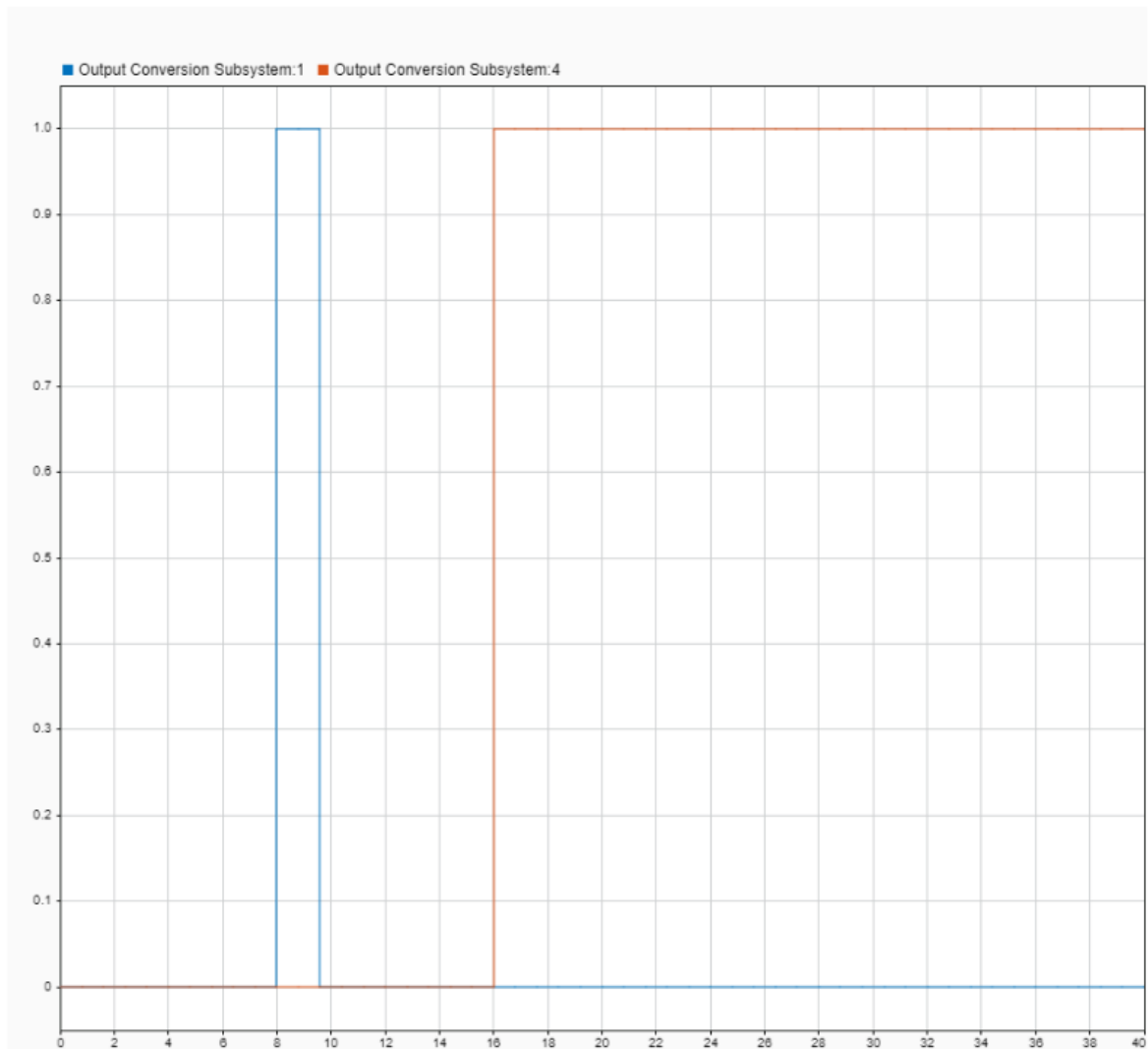


Ilustración 15: Resultados del test1

Como podemos observar, nos interesa que la señal *login\_correcto* (señal naranja) se mantenga igual a 1, lo que quiere decir que el usuario permanecerá con la sesión iniciada el tiempo que esté activo en la aplicación.

- **Test2:** En este test, la variable Selector es 2, por lo que directamente el usuario pasa al login, ya que en su momento ya se registró, por lo tanto aquí solamente veremos que la



variable `login_correcto` (señal naranja) toma el valor de 1, no el registro\_correcto (señal azul):

Step	Transition	Next Step
<b>Run</b> <pre>%% Initialize data outputs. Selector = 0; Correo_valido = false; Contrase_a_valida = false; Datos_validos = false; Codigo_correo = false;</pre>	1. <code>after(1,sec)</code>	step_1 ▼
<b>step_1</b> <pre>Selector = 2;</pre>	1. <code>after(1,sec)</code>	step_2 ▼
<b>step_2</b> <pre>Correo_valido = true;</pre>	1. <code>after(1,sec)</code>	step_3 ▼
<b>step_3</b> <pre>Contrase_a_valida = true;</pre>	1. <code>after(1,sec)</code>	step_4 ▼
<b>step_4</b> <pre>Codigo_correo = true; Selector = 0;</pre>		

Ilustración 16: Código el test2

#### Simulation Data Inspector: Inspect

Report Generated 07-Jan-2021 17:35:58

Run 7: Registro\_TEST\_b

Name	Line
Output Conversion Subsystem:1	—
Output Conversion Subsystem:4	—

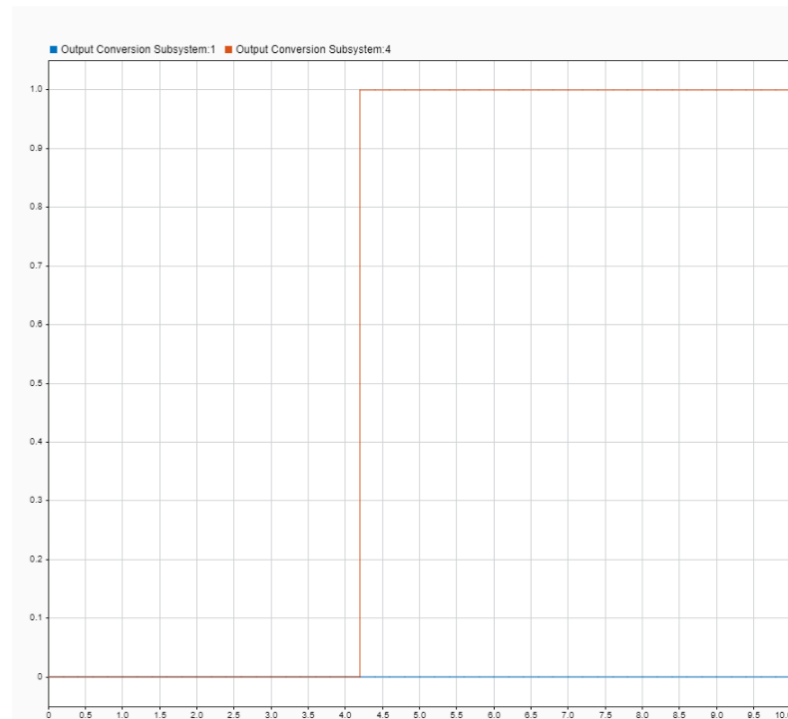


Ilustración 17: Login

- **Test3:** en este caso no se realiza ni el registro ni el login porque se produce un error, haciendo que la variable intentos llegue a su máximo, sacando un 1 por la señal de salida, en este caso de la señal de *error\_registro* (señal azul):

Step	Transition	Next Step
<b>Run</b>  %% Initialize data outputs. Selector = 0; Correo_valido = false; Contrase_a_valida = false; Datos_validos = false; Codigo_correo = false;	1. after(1,sec)	step_1 ▼
<b>step_1</b>  Selector = 1;	1. after(1,sec)	step_2 ▼
<b>step_2</b>  Correo_valido = true;	1. after(1,sec)	step_3 ▼
<b>step_3</b>  Contrase_a_valida = false;		

Ilustración 18: Test3 de error en el registro

Efectivamente, como vemos, es la contraseña la que no cumple con los requisitos de seguridad, manteniéndose a false todo el rato, hasta que la señal de error se activa, como vemos a continuación:

## Simulation Data Inspector: Inspect

Report Generated 07-Jan-2021 17:47:44

### Run 8: Registro\_TEST\_c

Name	Line
Output Conversion Subsystem:2	—
Output Conversion Subsystem:5	—

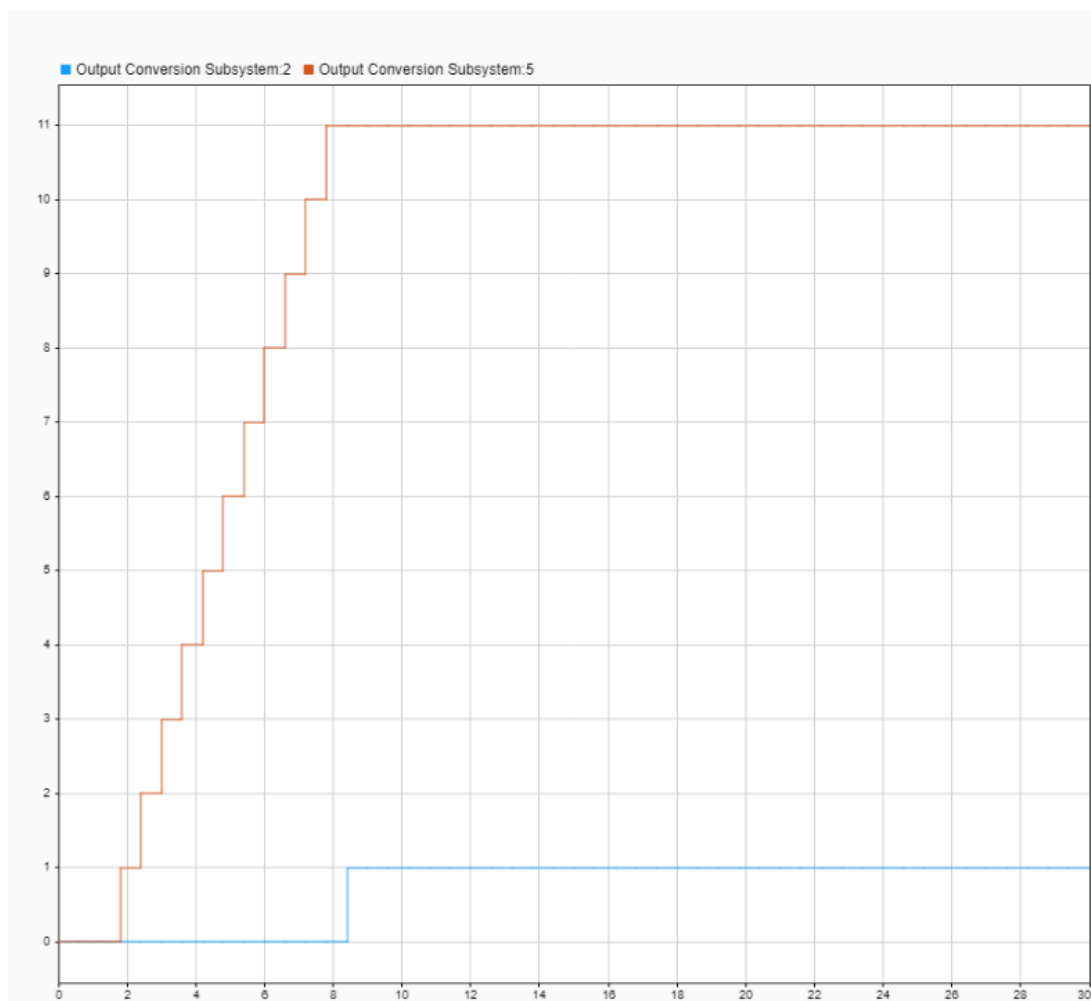


Ilustración 19: Señal de error con los intentos realizados

Como vemos, también se ha incluido la señal de los intentos que se realizan antes de que salte el error.

La implementación de este requisito también ha dejado margen de mejora. Por ejemplo, ha faltado la implementación de otro botón de cancelar cuando el usuario lo requiera, que no sería otra cosa que una señal más de entrada.

## RFU04- Añadir huerto

El objetivo principal de este modelo es proporcionar al usuario la capacidad de crear un nuevo huerto partiendo de un usuario previamente creado y habiendo realizado el 'login'. Además, dicho usuario deberá poseer los permisos necesarios para poder gestionar los huertos. Para ello, se ha realizado un esquema mediante Simulink:

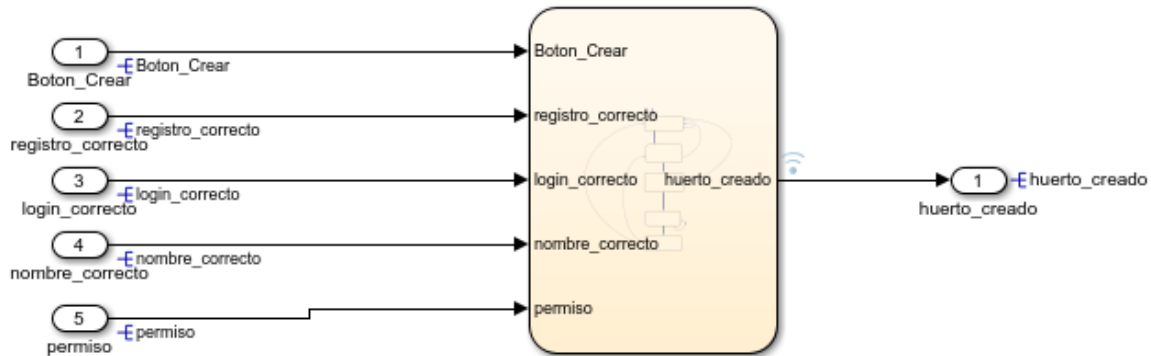


Ilustración 20: Esquema de entradas-salidas de añadir huerto

En primer lugar, se ha desarrollado un diagrama de estados que simula el proceso de creación de huerto dentro de la aplicación. Dicho diagrama podemos verlo a continuación:



<b>Run</b>  <pre>%% Initialize data outputs. Boton_Crear = true; registro_correcto = false; login_correcto = true; nombre_correcto = false; permiso=3;</pre>	1. <code>after(3,sec)</code>	Compr... ▼	Se inicializa
<b>Comprobar_usuario</b> <pre>registro_correcto=true;</pre>	1. <code>after(1,sec)</code>	Compr... ▼	Se comprueba que el usuario está registrado
<b>Comprobar_permiso</b>	1. <code>true</code>	Introdu... ▼	Se comprueba que el usuario posee permiso
<b>Introducir_nombre</b> <pre>nombre_correcto=true;</pre>	1. <code>true</code>	Huerto... ▼	Se comprueba que el nombre no está en uso
<b>Huerto_creado</b> <pre>Boton_Crear=false</pre>			Se crea el huerto.

Ilustración 22: Código de test de creación de huerto

En la representación de la variable “*huerto\_creado*” podemos observar cómo se torna en verdadero, permitiendo al usuario haber creado el huerto.

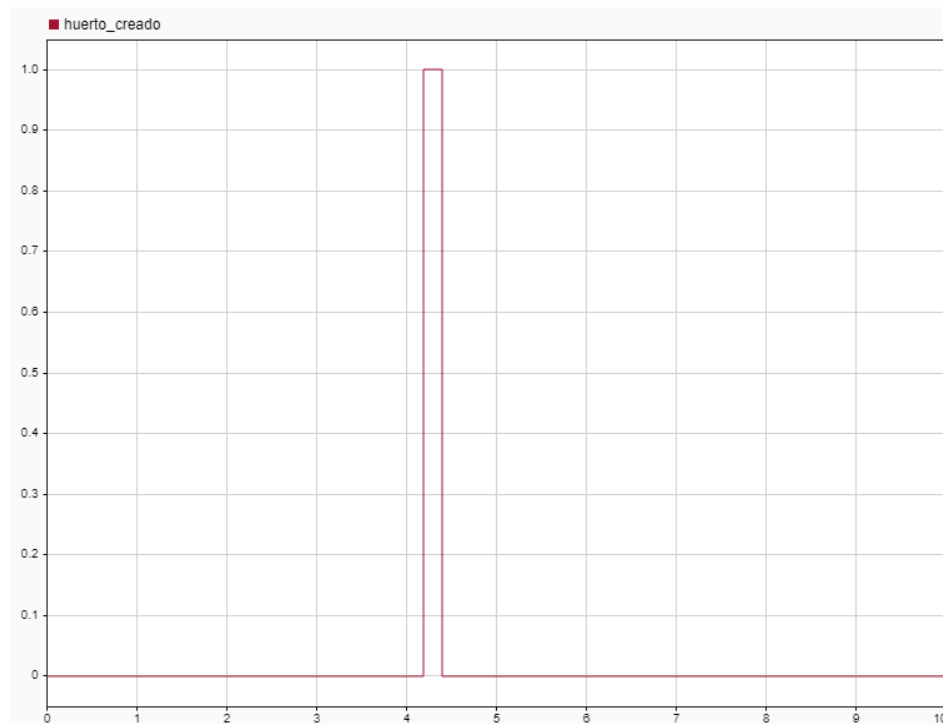


Ilustración 23: Resultado del test de creación de huerto.

## RNFU03- Opción de control manual

En este caso, se va a comprobar la capacidad del sistema para ofrecer control manual de los actuadores al usuario. Según el documento de requisitos, dicho control se podrá ofrecer en dos casos particulares: cuando exista un fallo en el control automático de los mismos o cuando el usuario lo solicite de forma expresa. Para ello, se ha realizado el siguiente modelo en Simulink:

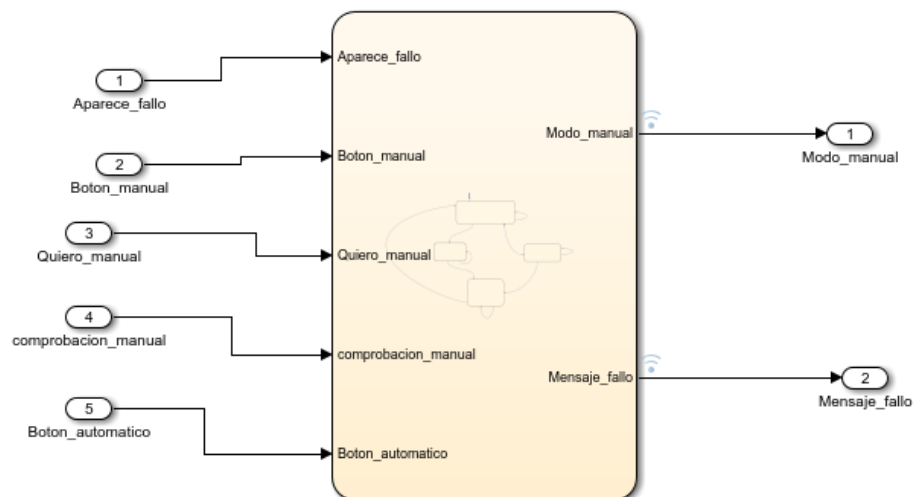


Ilustración 24: Esquema de conexiones entrada-salida del modelo de actuadores en modo manual/automático.

De nuevo, hemos creado un diagrama de estados que define el comportamiento del sistema para su configuración en modo manual.

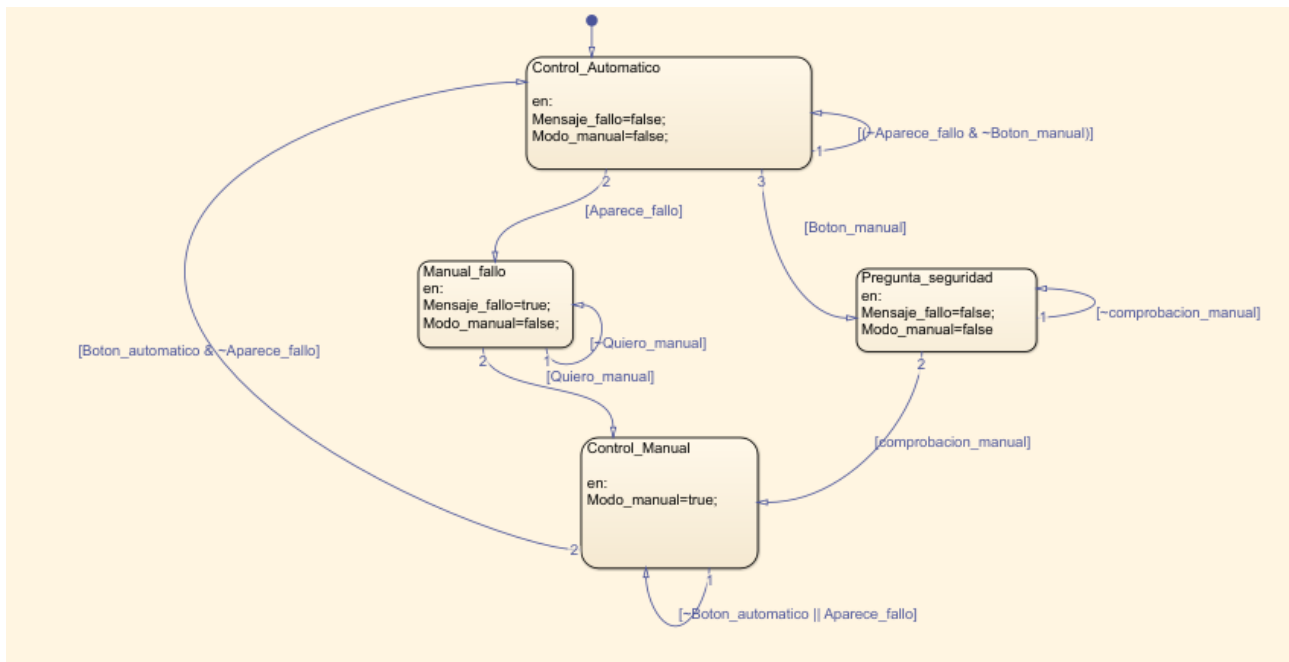


Ilustración 25: Diagrama de conexión de estados para el sistema de actuadores en modo manual/automático.

Por otro lado, ya que existen dos formas de configurar el sistema en modo manual, se han establecido dos test por los cuales se comprueba el funcionamiento de estas.

- **Test 1:** Se comprueba en este primer test, la capacidad de activar el modo manual de los actuadores de forma intencionada a través de la aplicación. Para ello, el usuario deberá responder a la pregunta afirmando que verdaderamente quiere el cambio. Esto permite establecer el control de tipo manual durante un periodo de tiempo limitado.



<b>Run</b> <pre>%% Initialize data outputs. Aparece_fallo = false; Boton_manual = true; Quiero_manual = false; comprobacion_manual = false; Boton_automatico = false;</pre>	1. true	pregun... ▼	Se inicializan las variables.
<b>pregunta_seguridad</b> <pre>comprobacion_manual=true; Boton_manual=true;</pre>	1. after(1,sec)	Contro... ▼	Se pregunta al usuario si quiere el cambio a manual.
<b>Control_manual</b> <pre>comprobacion_manual=false; Boton_manual=false;</pre>	1. after(6,sec)	Contro... ▼	Se establece el control manual durante un periodo.
<b>Control_automatico</b> <pre>Boton_automatico=true;</pre>			volvemos a automático

Ilustración 26: Código para el test1 de modo manual/automático.

Con esta prueba, obtenemos los siguientes resultados:

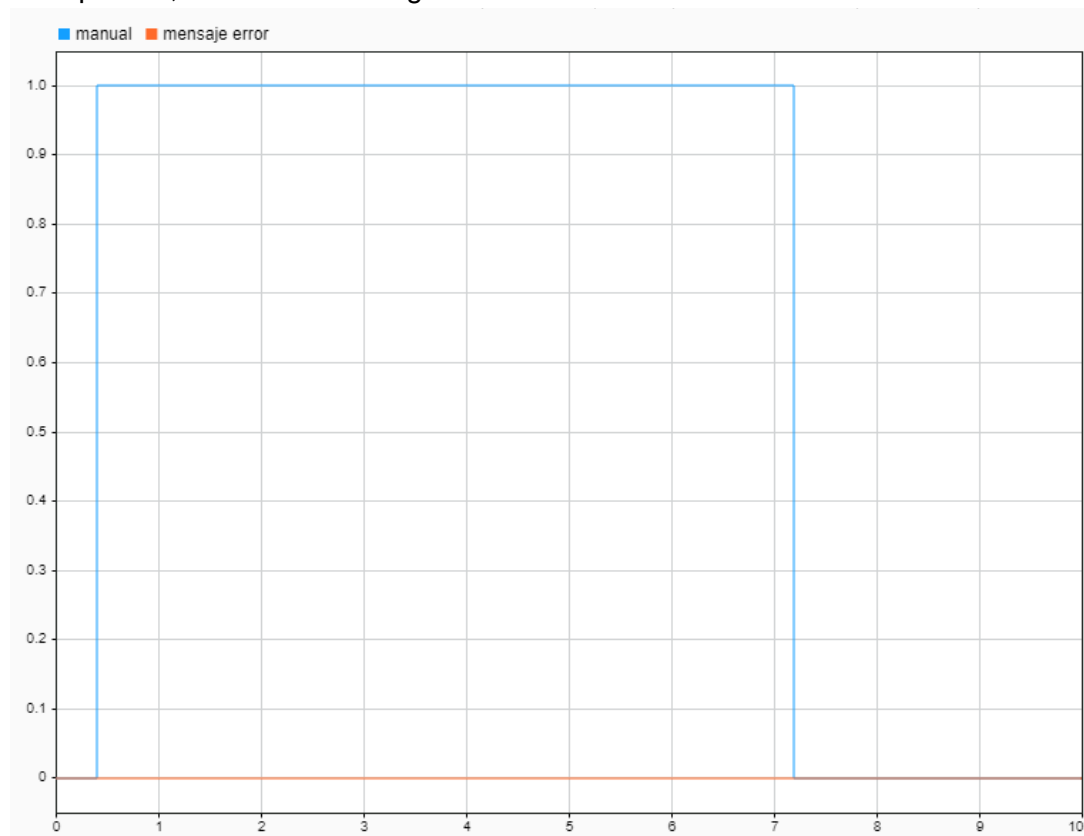


Ilustración 27: Resultado del test 1 de modo manual/automático.

Se observa como el control manual se ha activado de forma satisfactoria durante un tiempo limitado.

- **Test 2:** En el segundo de los test se ha comprobado el funcionamiento cuando existe un mensaje de error por parte del modo automático de funcionamiento. Tras aparecer el error, se pregunta al usuario si quiere activar el modo manual y, tras su aceptación, se activará dicho modo hasta solucionar el error.

Step	Transition	Next Step	Description
<b>Run</b>  <pre>%% Initialize data outputs. Aparece_fallo = true; Boton_manual = false; Quiero_manual = false; comprobacion_manual = false; Boton_automatico = false;</pre>	1. <a href="#">after(1,sec)</a>	Manua... ▼	Se inicializan las variables.
<b>Manual_fallo</b> Quiero_manual= <a href="#">true</a> ;	1. <a href="#">after(3,sec)</a>	Contro... ▼	Tras aparecer el fallo, el usuario confirma.
<b>Control_manual</b> Boton_automatico= <a href="#">true</a> ;	1. <a href="#">after(5,sec)</a>	Arregl... ▼	Se establece el modo manual
<b>Arreglo_fallo</b> Aparece_fallo= <a href="#">false</a> ;			Se soluciona el fallo, volviendo a modo automático.

Ilustración 28: Código del test 2 de modo manual/automático.

Los resultados obtenidos de la prueba los podemos observar en la siguiente imagen:

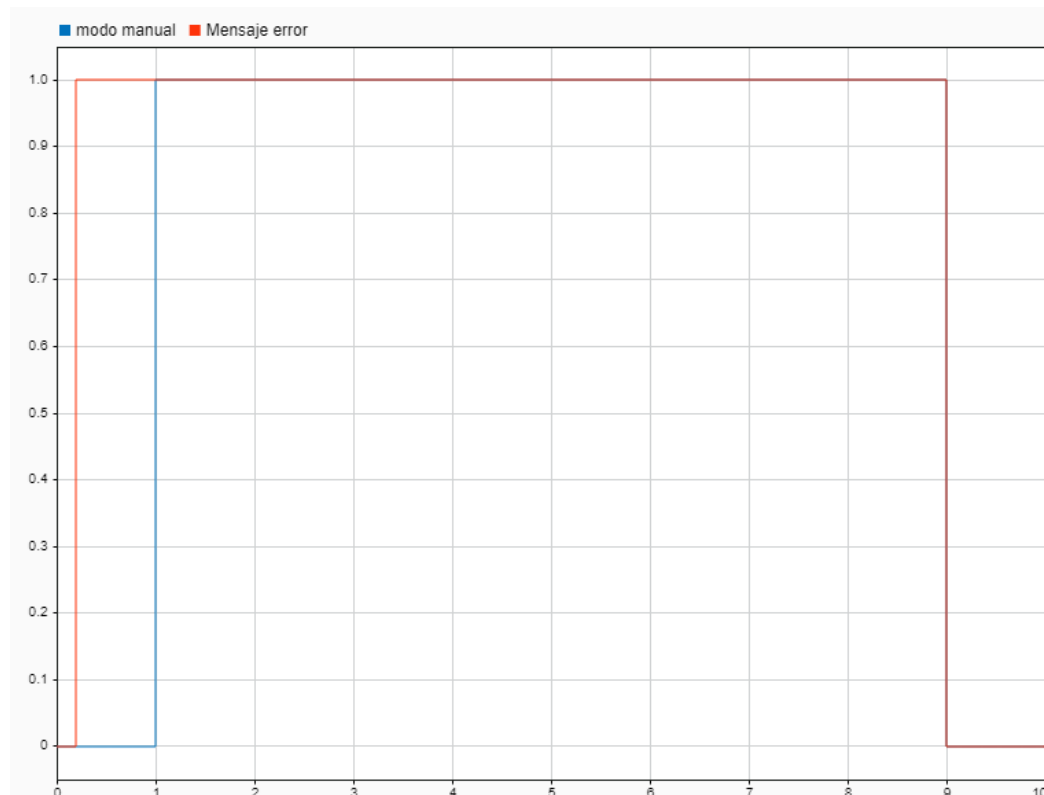


Ilustración 29: Resultado del test 2 de modo manual/automático.

Se comprueba que el sistema responde tal y como era de esperar en el test.

## RNFME01- Fallo en el riego

Para la verificación de este requisito, se ha elaborado el modelo de lo que sería la planta real del sistema de riego que tendría uno de los huertos. Para ello se ha hecho uso de Simulink como herramienta de desarrollo.



Ilustración 30. Modelo de Riego

Este dispone de una entrada, que se trata de la actuación del riego. Cuenta además con dos salidas, que son los valores de humedad del terreno y una variable que verifica el correcto funcionamiento del sistema de riego.

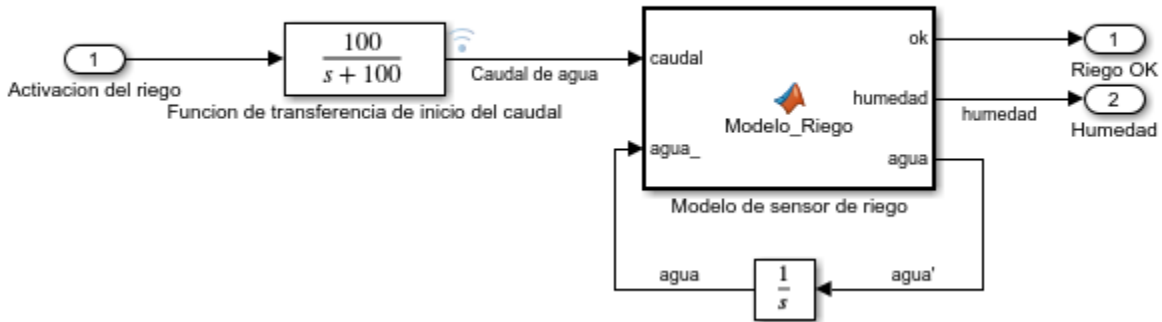


Ilustración 31. Detalles del modelo de riego

Dentro de este podemos encontrar este diagrama el cual nos genera un caudal a la activación de riego, que hará acumularse el agua en el sistema y aumentar la humedad de salida del sistema.

Para la verificación de si el sistema de riego está funcionando de manera correcta, se ha dispuesto que, con las medidas de humedad del terreno, se haga una verificación inicial, haciendo que el sistema empiece el riego y el sensor de humedad en el terreno mida si está aumentado esta. Si la respuesta del sistema de riego y los valores de humedad son adecuados, la variable “Riego OK” nos deberá dar una respuesta verdadera y viceversa.

Step	Transition	Next Step	Description
Run Activacion_riego = 0;	1. after(1,sec)	step_1 ▼	Se inicializa con el riego desactivado
: step_1 Activacion_riego = 1;	1. after(1,sec)	step_2 ▼	Se activa el riego durante 1 segundo
step_2 Activacion_riego = 0;			Se desactiva el riego

Ilustración 32. Banco de pruebas para el sistema de riego

Para los test de este requisito se ha trabajado en el mismo banco de pruebas, pero usando un modelo diferente de sistema de riego (uno correcto y otro erróneo). Este banco se muestra a continuación:

Para la verificación de este requisito se han elaborado dos test, que se desarrollarán en las siguientes líneas:

### - **Test 1:**

El primer test se realizará en una planta que funciona de manera correcta, y el sistema debe dar una respuesta positiva de que el sistema está trabajando de manera correcta. Para ello se debe verificar la siguiente secuencia de eventos:

► At any point of time, if Activacion\_riego > 0 becomes true then, with a delay of at most 5 seconds, Riego\_OK > 0 must be true

A continuación, se muestra el resultado de realizar el test con el banco de pruebas propuesto.

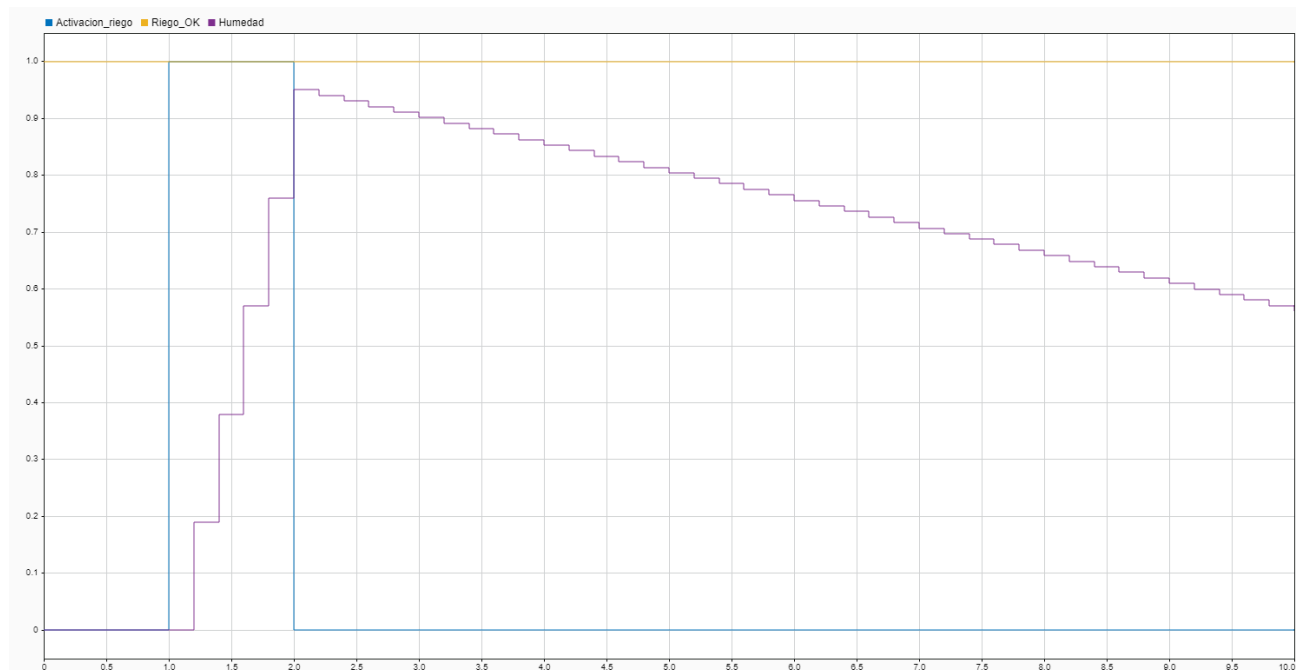


Ilustración 33. Resultado del test 1 de fallo de riego

Se puede observar cómo se pasa de forma correcta el test, consiguiendo reportar de forma adecuada que el sistema de riego funciona de una manera correcta y sin fallos.

### - **Test 2:**

El segundo test se realizará de la misma manera que con el test 1, pero con una planta que su funcionamiento no es el adecuado, y por tanto a pesar de darle la acción de que empiece el riego, el dispositivo de riego no consigue regar el huerto y por tanto aumentar la humedad de la tierra. Para la verificación de este test, se debe dar la siguiente secuencia de eventos:

► At any point of time, if Activacion\_riego > 0 becomes true then, with a delay of at most 5 seconds, Riego\_OK < 0.5 must be true

A continuación, se puede ver el resultado de este test.

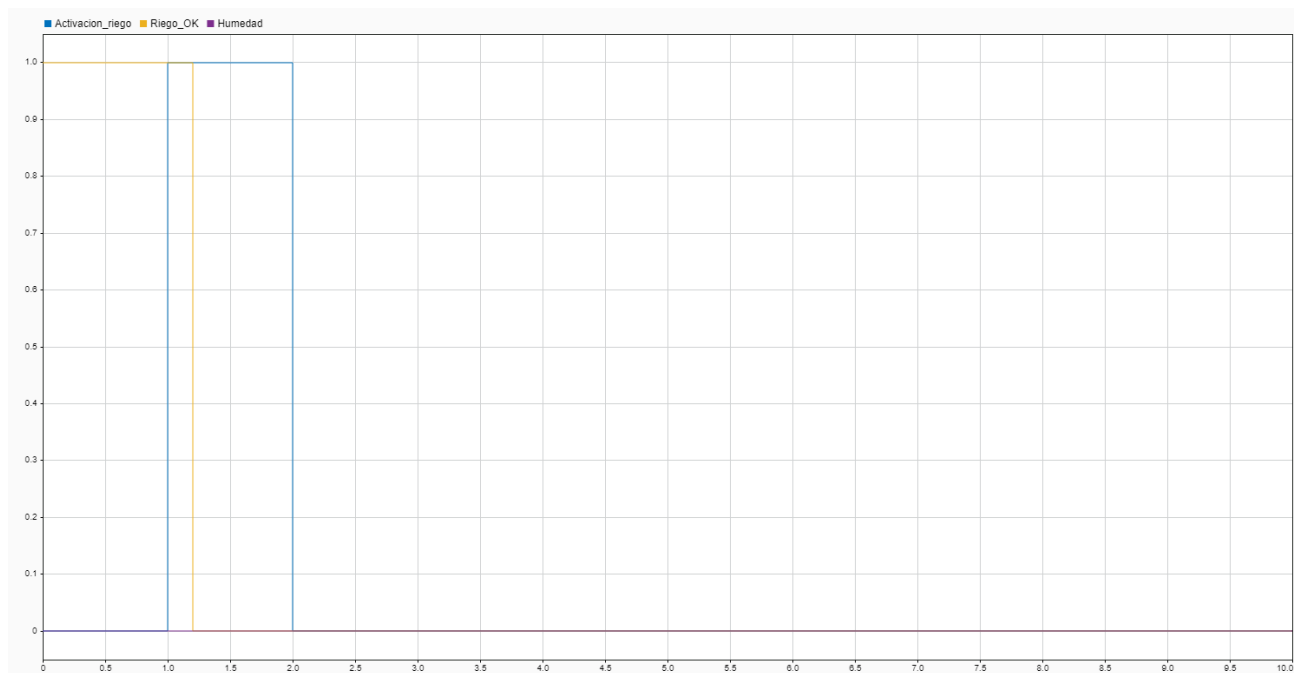


Ilustración 34. Resultado del test 2 de fallo de riego

En este caso se puede observar, que aun dando una acción de que el sistema empiece a regar, la humedad del huerto no aumenta, por lo tanto, no está funcionando de manera adecuada este actuador, y el sistema es capaz de reportarlo de forma adecuada.

## RNFD01 – Velocidad de actuación de los actuadores

Para la verificación de este requisito se empleará el mismo modelo y banco de pruebas que le propuesto para verificar la existencia de fallo en el sistema de riego (RNFME01). Es por ello que la parte introductoria de explicación se omitirá y se pasará a la explicación del test realizado para la verificación de este requisito.

### - **Test 1:**

El test que se realizará consistirá en la activación de la actuación del riego y se deberá comprobar cuanto tiempo ha tardado en darse la actuación solicitada, cumpliendo que sea menor a la establecida por el requisito. Esto queda recogido en la siguiente secuencia de eventos:

► At any point of time, if `Activacion_riego == 1` becomes true then, with a delay of at most **1 seconds**, `Caudal>0` must be true

Y tras la realización de este test, se obtiene el siguiente resultado:

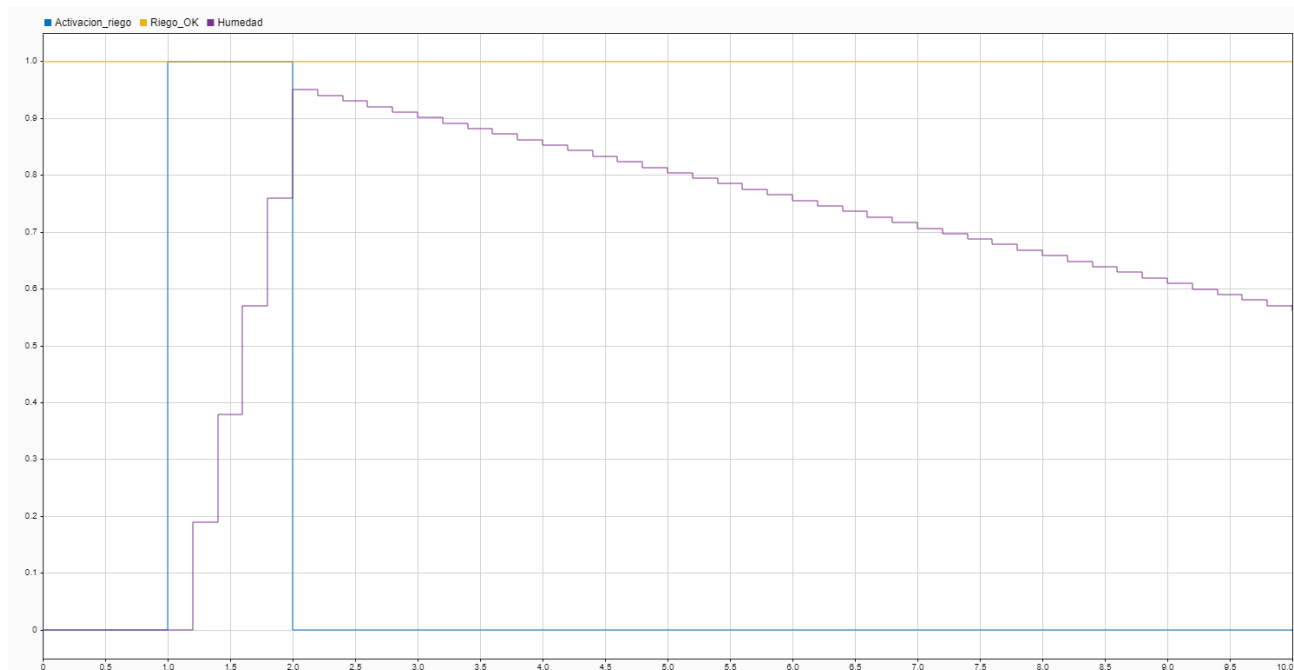


Ilustración 35. Resultados del test 1 de velocidad de actuación de los actuadores

Por lo tanto, los resultados obtenidos por el test son positivos y queda verificado este requisito.

## RNFD02 – Recogida de información de los sensores

Para la verificación de este requisito, se ha realizado un modelo con la herramienta Simulink que simula el comportamiento de los sistemas reales de sensorización que estarán dispuestos en nuestro huerto y que serán necesarios para el correcto funcionamiento de la aplicación.

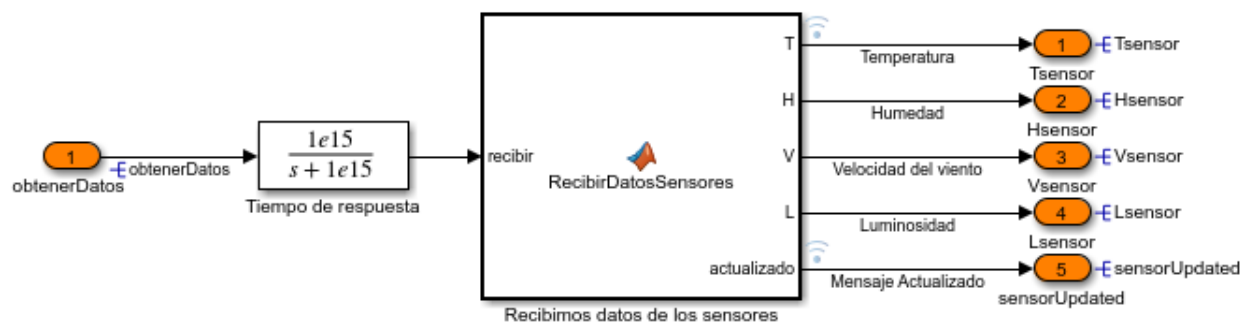


Ilustración 36. Modelo de la sensorización del huerto

Este modelo dispone de una entrada, la cual se encarga de solicitar actualizar los valores de los sensores del sistema. Disponemos de cinco salidas, las cuales corresponden cuatro a los valores de los sensores (Temperatura, Humedad, Velocidad del viento y Luminosidad) y uno a la confirmación de que se han actualizado los valores de estos sensores de manera correcta.

Se pide que el sistema los sensores han de devolver los datos en un tiempo suficientemente rápido tras solicitar la medición de los sensores. Para ello se ha elaborado un banco de pruebas donde se pide una actualización de los datos cada 0,2 segundos y se esperan 2 segundos a que los datos lleguen al sistema.

Step	Transition	Next Step	Description
Run obtenerDatos = 0;	1. after(0.1,sec)	step_1 ▼	Inicializacion
step_1 obtenerDatos = 1;	1. after(2,sec)	step_2 ▼	Se piden datos
step_2 obtenerDatos = 0;	1. after(0.1,sec)	Run ▼	Se resetea

Ilustración 37. Banco de pruebas del modelo de sensorización del huerto

A continuación, se desarrollará el test que se ha realizado para verificar este requisito.

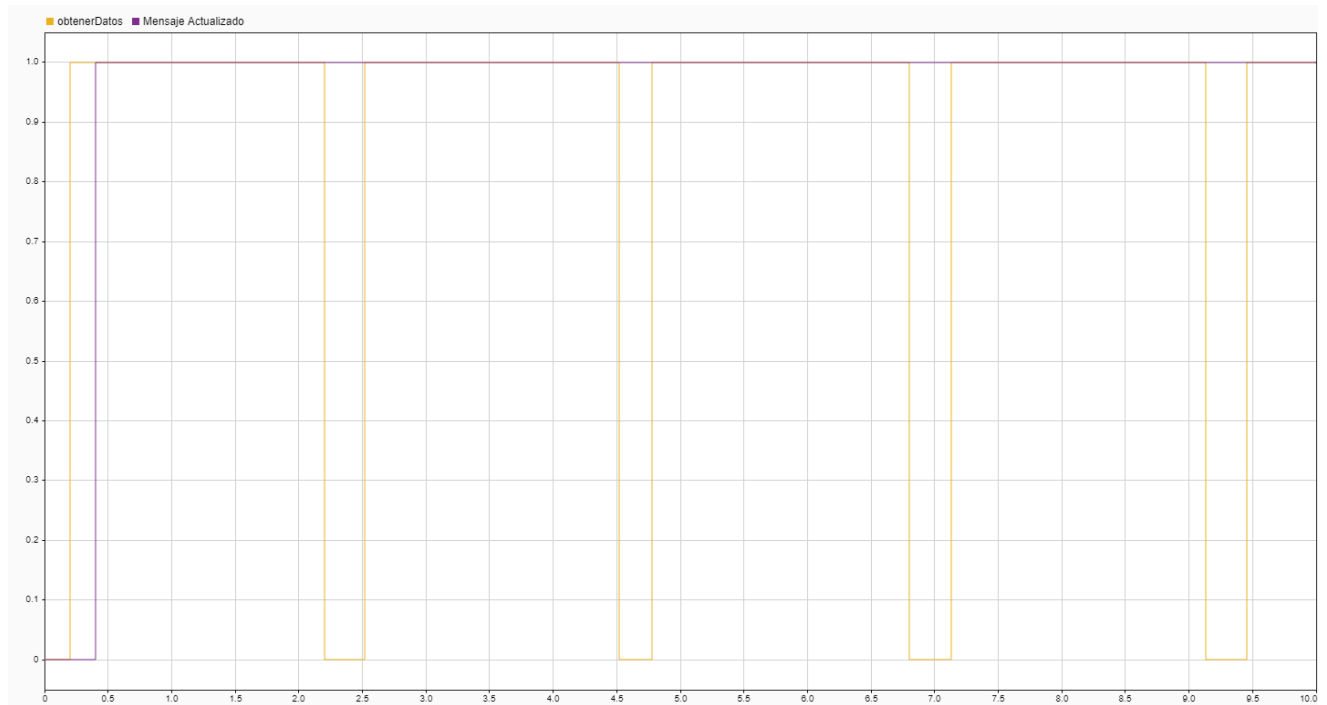
- **Test 1:**

Este test consistirá en la comprobación de la velocidad de respuesta de los datos solicitados y verificando que esta se cumple. Esto queda registrado en la siguiente secuencia de eventos:

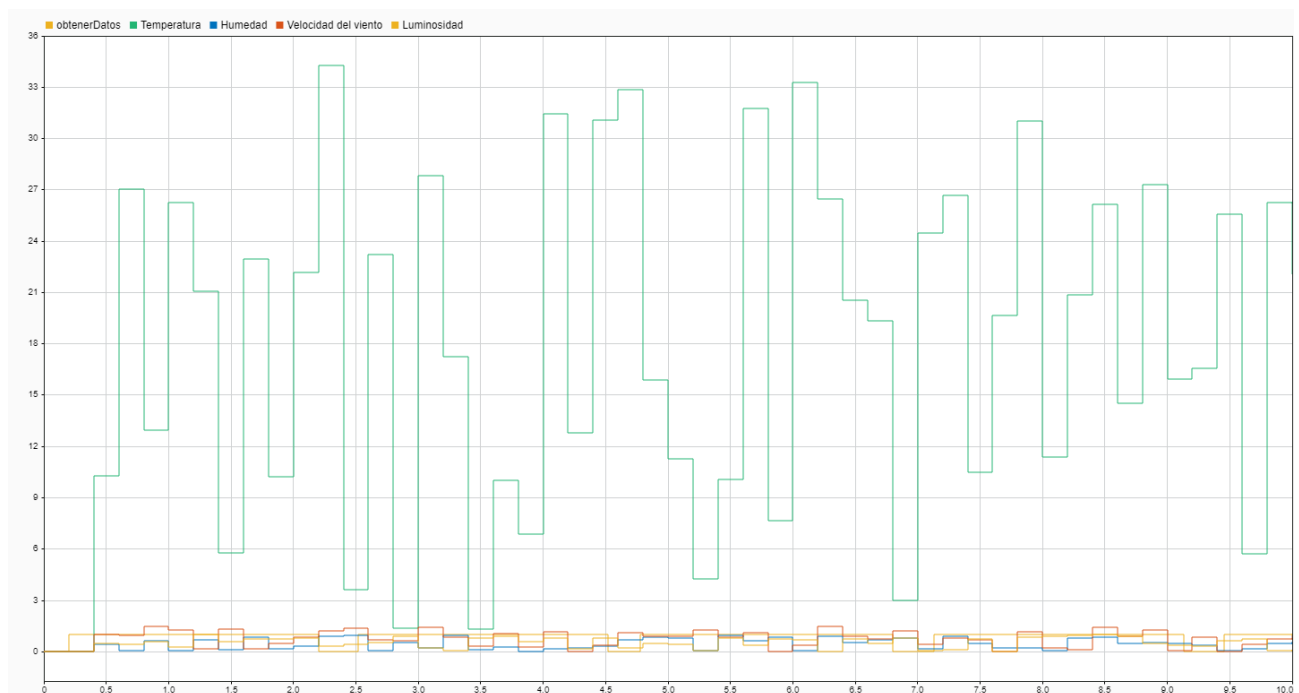
► At any point of time, if obtenerDatos > 0 becomes true then, with a delay of at most 30e-3 seconds, datosActualizados > 0 must be true

Realizando este test se obtiene el siguiente resultado:





**Ilustración 38. Resultado del test 1 de recogida de información de sensores con la señal de mensaje actualizado**



**Ilustración 39. Resultado del test 1 de recogida de información de sensores con los valores recogidos por los sensores**

Se puede observar que los requisitos establecidos en este requisito son demasiado exigentes, y nuestro sistema no cumple estos test debido a la frecuencia de muestreo que funciona nuestro sistema. A partir de aquí se tienen dos alternativas para superar este test, bajar las exigencias del requisito o montándose en un sistema con mayor velocidad de muestreo si es realmente lo que nuestro sistema exige para su correcto funcionamiento.

## RNFME03 – Fallo en los sensores

Para la verificación de este requisito se ha se comparte el modelo ya presentado para la verificación del requisito de recogida de información de sensores (RNFD02), ya que se realiza sobre la misma parte del sistema que el sistema de sensorización del huerto. También se desarrollará sobre el mismo banco de trabajo desarrollado durante la verificación de ese requisito, por lo que se pasará directamente al desarrollo de los test elaborados para la verificación de este requisito.

### - **Test 1:**

Este test se desarrollará sobre el sistema de sensorización que tiene un comportamiento de forma correcta, y se debe comprobar que el sistema reporta que los datos han sido actualizados de forma adecuada. Para ello se ha realizado el test con la siguiente secuencia de eventos:

► At any point of time, if `obtenerDatos > 0` becomes true then, with a delay of at most 2 seconds, `datosActualizados > 0` must be true

Ejecutando este test se obtiene el siguiente resultado:

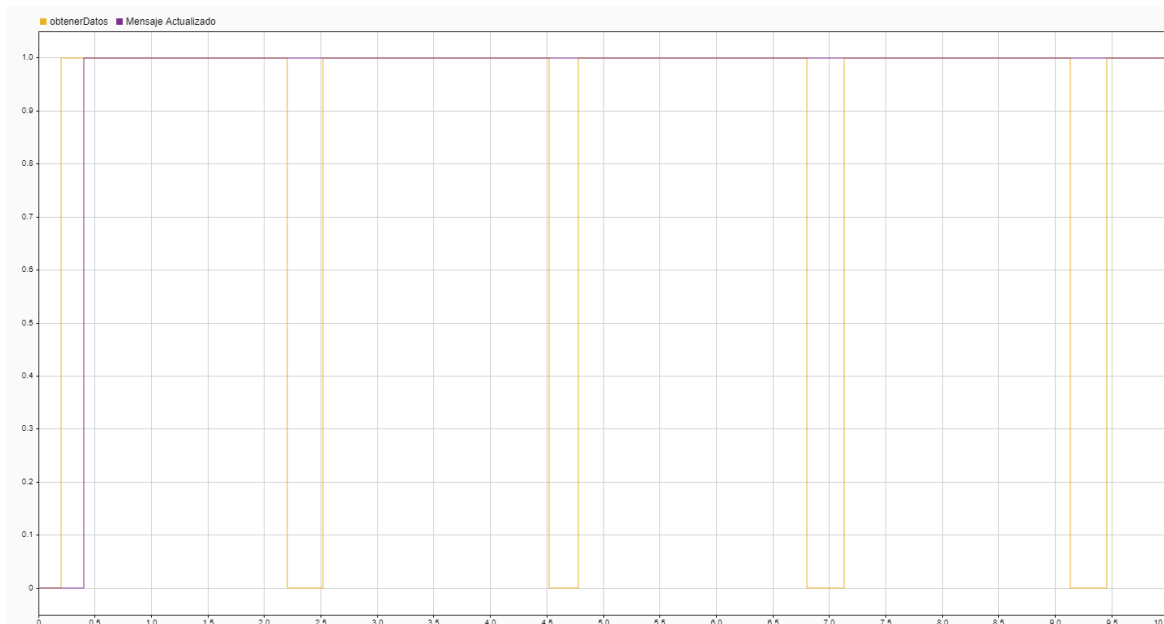


Ilustración 40. Resultado del test 1 de fallo en los sensores

Se puede observar cómo los resultados del test son positivos y se obtiene que el sistema actualiza y reporta el correcto funcionamiento del sistema.

- Test 2:

Para este test, se realizará la prueba con un sistema que tiene un funcionamiento no adecuado, no dando en consecuencia datos adecuados los valores de salida de los sensores y teniéndose que reportar que aun pidiendo que se actualicen estos datos, los valores de los sensores no han sido actualizados de forma adecuada, desechando los valores obtenidos. Para este test se han verificado la siguiente secuencia de eventos:

► At any point of time, if obtenerDatos > 0 becomes true then, with a delay of at most 2 seconds, datosActualizados == 0 must be true

Y ejecutando este test se obtiene el siguiente resultado:

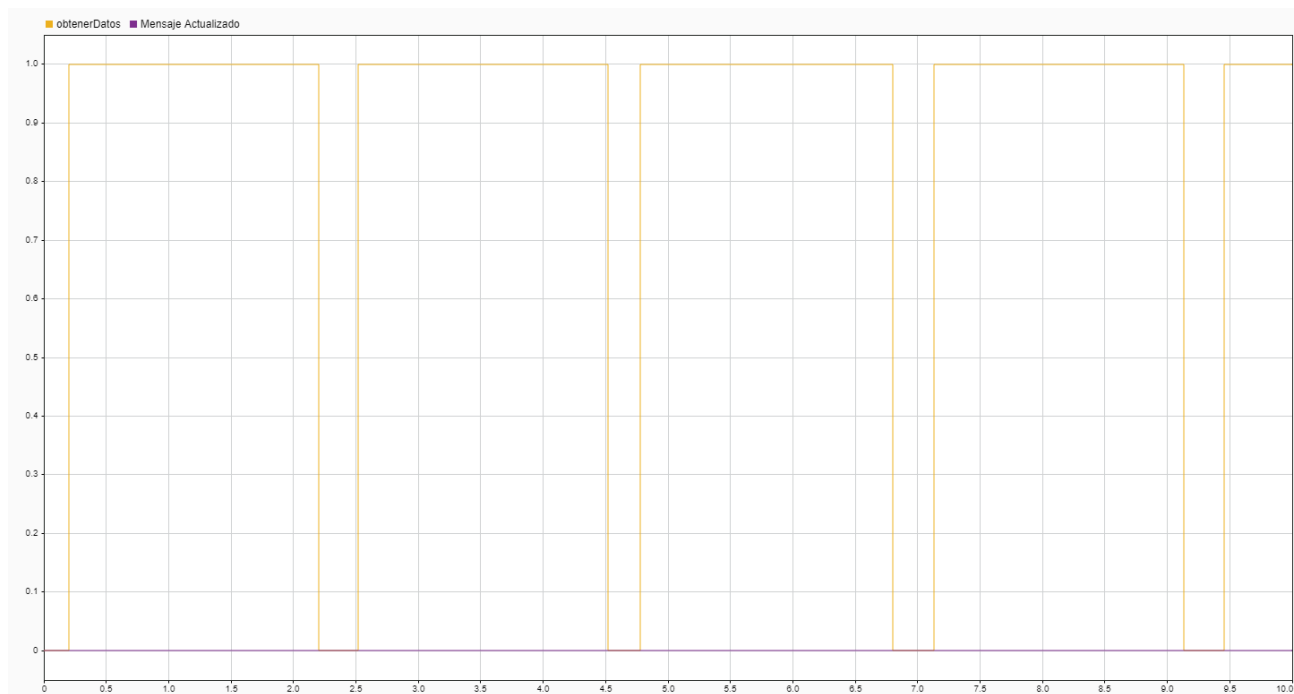


Ilustración 41. Resultado del test 2 de fallo en los sensores

Se puede observar como de forma correcta, se reporta que el sistema no está devolviendo los valores adecuados de los sensores y que por lo tanto el sistema debe advertir que los sensores no se han actualizado de forma adecuada. Es por ello por lo que de esta manera queda verificado este requisito.

## RNFD04- Conexión con parte meteorológico

Las siguientes pruebas realizadas son sobre el requisito de comunicación meteorológica para adquirir los datos necesarios para poder activar o no los actuadores en función de los datos obtenidos por los sensores.



Ilustración 42: Modelo de la conexión con parte meteorológico

Este requisito determina que solo recogeremos la información en tres situaciones diferentes. Dos de ellas serán cuando pase un día entero o media hora, es decir, sin necesidad de petición por parte del usuario. La otra, entonces, ese cuando el usuario lo pida.

Para este requisito se han realizado dos test diferentes:

### **-Test 1:**

El primero de todos evaluaremos las situaciones que pase un día entero o que se pida actualizar por parte del usuario. Y comprobaremos que efectivamente se ofrecen los datos al pedirse esta actualización.

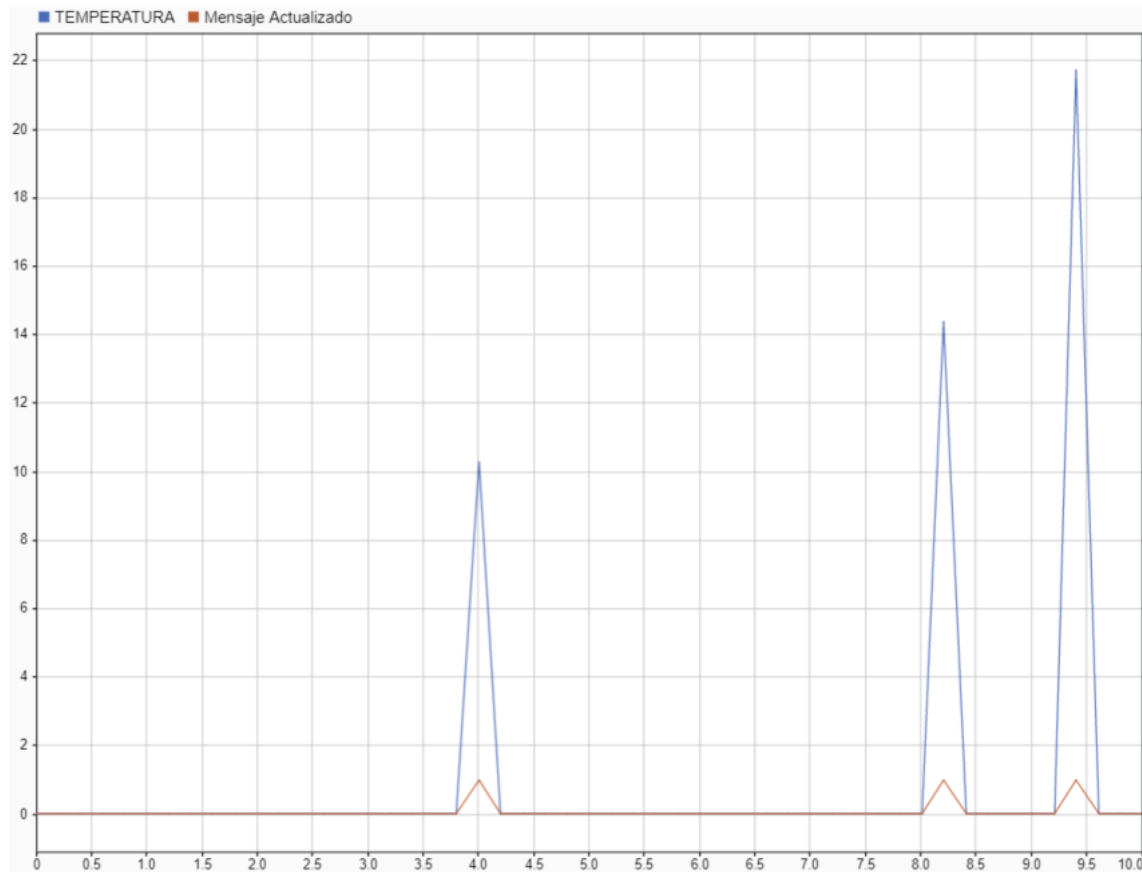


Ilustración 43: Resultado del test 1 conexión con parte meteorológico

Observamos como se han tratado de simular dos situaciones, en primer lugar, el cambio de día a los 4 segundos, otro cambio de día después y más tarde, en un momento cualquiera, el usuario trata de actualizar por su cuenta.

Step	Transition	Next Step
<b>Run</b> %% Initialize data outputs. hora = 0; actualiza = 0;	1. after(4,sec)	step_1 ▼
<b>step_1</b> hora = 24; actualiza = 0;	1. true	step_2 ▼
<b>step_6</b> hora=0; actualiza= 0;	1. after(4,sec)	step_5 ▼
<b>step_5</b> hora = 24; actualiza = 0;	1. true	step_2 ▼
<b>step_2</b> hora = 0; actualiza = 0;	1. after(1,sec)	step_3 ▼
<b>step_3</b> hora=0; actualiza=1;	1. true	step_4 ▼
<b>step_4</b> hora=0; actualiza=0;		

Ilustración 44: Banco de pruebas test 1 parte meteorológico

### -Test 2:

El segundo test se ha tratado de complicar un poco más. En él no sólo incluiremos la condición de que pase un día, sino que también la de 30 minutos, además de que el usuario pueda pedir que se actualice en cualquier momento. Observaremos las salidas tanto de la temperatura, teniendo en cuenta que de la misma manera se obtendrían el resto de datos pero por simplificar el gráfico no se han mostrado y el mensaje que obtendría el usuario de actualización del parte meteorológico.

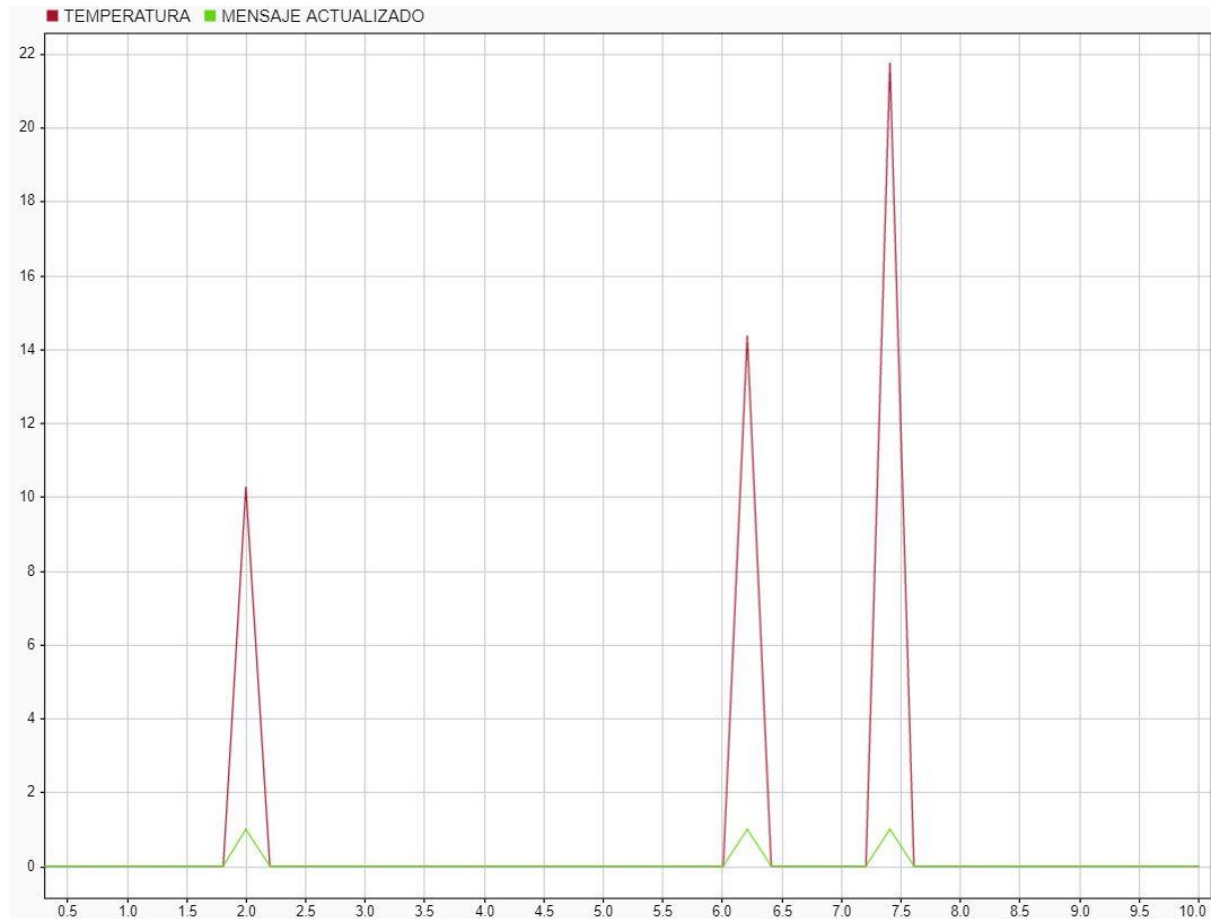


Ilustración 45: Resultado test 2 conexión parte meteorológico

Podemos observar cómo en primer lugar, después de 2 segundos (simulando que pasan 30 minutos) actualiza el parte meteorológico. Después a los cuatro segundos tenemos el cambio de día y finalmente un segundo más tarde tenemos la actualización por parte del usuario.

Step	Transition	Next Step	Description
Run %% Initialize data outputs. hora = 0; actualiza = 0; minutos = 0;	1. after(2,sec)	step_1 ▼	que pasa media hora
step_1 hora=0; actualiza=0; minutos=30;	1. true	step_2 ▼	: sucede actualizamos
step_2 hora=0; actualiza=0; minutos=0;	1. after(4,sec)	step_3 ▼	imula que pasa un día
step_3 hora=24; actualiza=0; minutos=0;	1. true	step_4 ▼	: sucede actualizamos
step_4 hora=0; actualiza=0; minutos=0;	1. after(1,sec)	step_5 ▼	ario decide actualizar
step_5 hora=0; actualiza=1; minutos=0;	1. true	step_6 ▼	Actualizamos
step_6 hora=0; actualiza=0; minutos=0;			

Ilustración 46: Banco de pruebas conexión parte meteorológico

## RNFD05- Permiso nivel usuario

En este segundo modelo simularemos el requisito de obtener permisos suficientes como para poder actuar sobre una aplicación del huerto, qué ocurre cuando no tienes los permisos de usuario suficientes como para poder actualizar esa aplicación y uniremos ambos modelos, la obtención de permiso y pedir parte meteorológico para simular las diferentes situaciones de forma mucho más real.



Ilustración 47: Modelo Permiso nivel usuario

### -Test 1:

En primer lugar, simularemos únicamente si el usuario que quiere acceder a una funcionalidad de la aplicación tiene o no permisos suficientes. Como se advirtió en los requisitos hay tres niveles de usuario en la obtención de permisos, por ello, de la misma

manera habrá tres tipos de funcionalidades dentro de las diferentes aplicaciones que hay en el huerto.

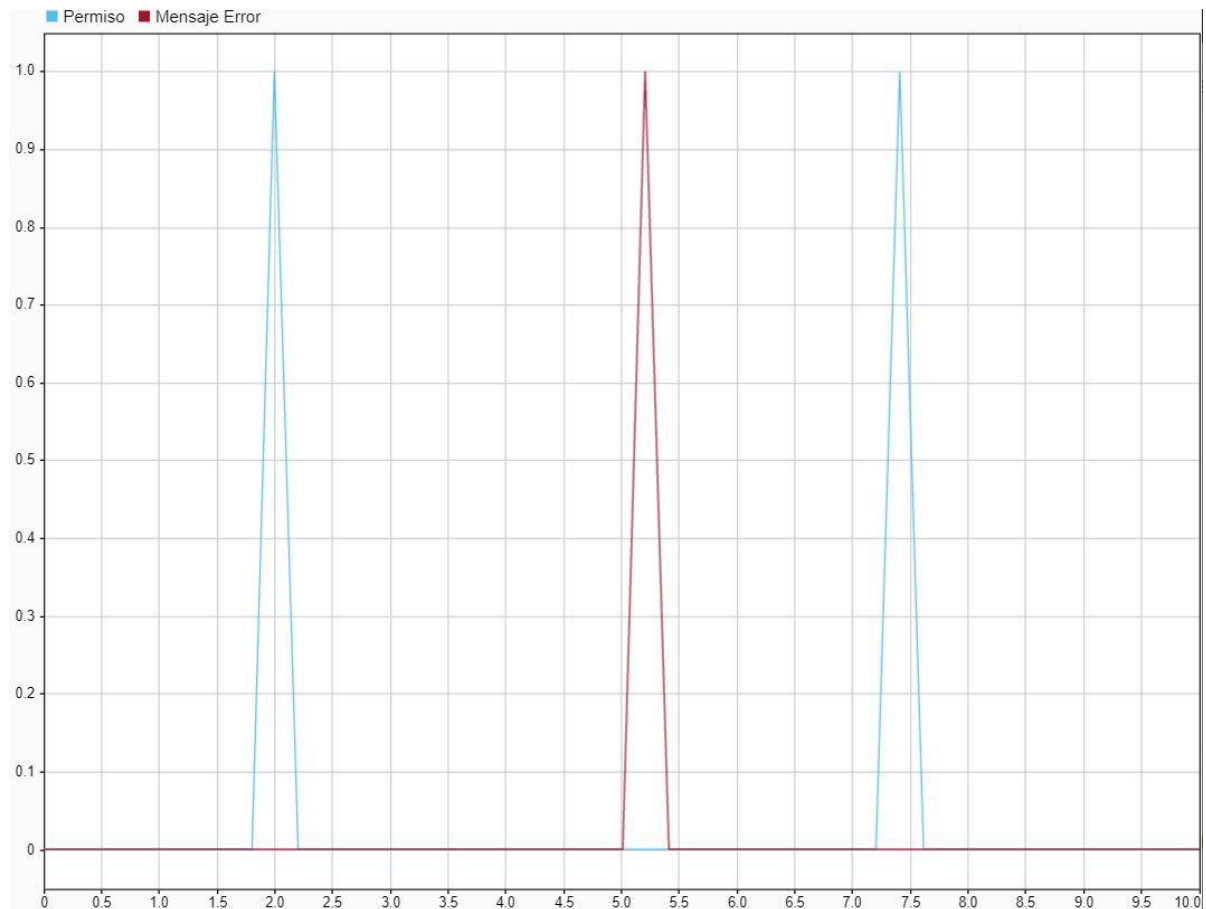


Ilustración 48: Resultado test 1 permiso nivel usuario

Observamos como en primer lugar, tendremos un nivel de usuario 1, el cual es el que tiene acceso a todas las funcionalidades y como se pide actuar sobre una situación de funcionalidad 2 se le da acceso. En cambio, segundos más tarde, tenemos un nivel de usuario 3 con una actuación de funcionalidad 2, lo que significa que no va a poder actuar sobre lo que quería y obtendrá un mensaje de error. Finalmente, un usuario de nivel 2 si podrá actuar sobre una aplicación de funcionalidad 3.



Step	Transition	Next Step
Run %% Initialize data outputs. nivelusuario = 0; funcionalidad = 0;	1. after(2,sec)	step_1 ▼
step_1 nivelusuario = 1; funcionalidad = 2;	1. true	step_2 ▼
step_2 nivelusuario = 0; funcionalidad = 0;	1. after(3,sec)	step_3 ▼
step_3 nivelusuario = 3; funcionalidad = 2;	1. true	step_4 ▼
step_4 nivelusuario = 0; funcionalidad = 0;	1. after(2,sec)	step_5 ▼
step_5 nivelusuario = 2; funcionalidad = 3;	1. true	step_6 ▼
step_6 nivelusuario = 0; funcionalidad = 0;		

Ilustración 49: Banco de pruebas permiso nivel usuario

### -Test 2:

Para esta segunda prueba, se unen tanto el modelo de obtención del parte meteorológico como el de obtención de permiso. Se ha determinado que la funcionalidad de pedir actualización del parte meteorológico es de nivel 2 y los resultados obtenidos son los siguientes.

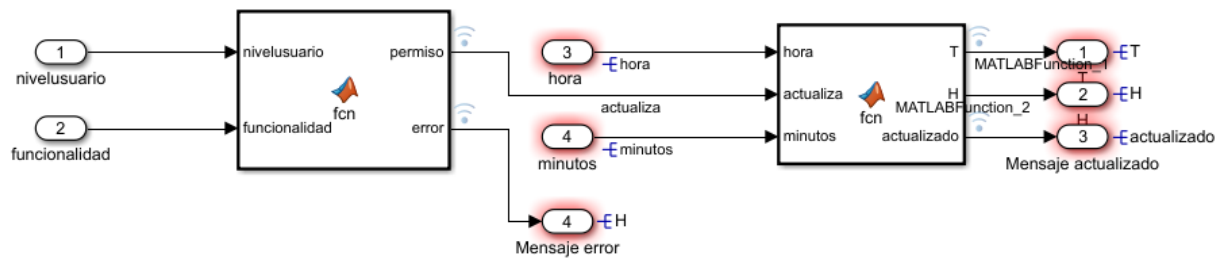


Ilustración 50: Modelo permiso nivel usuario y conexión parte meteorológico

Los resultados son los siguientes:

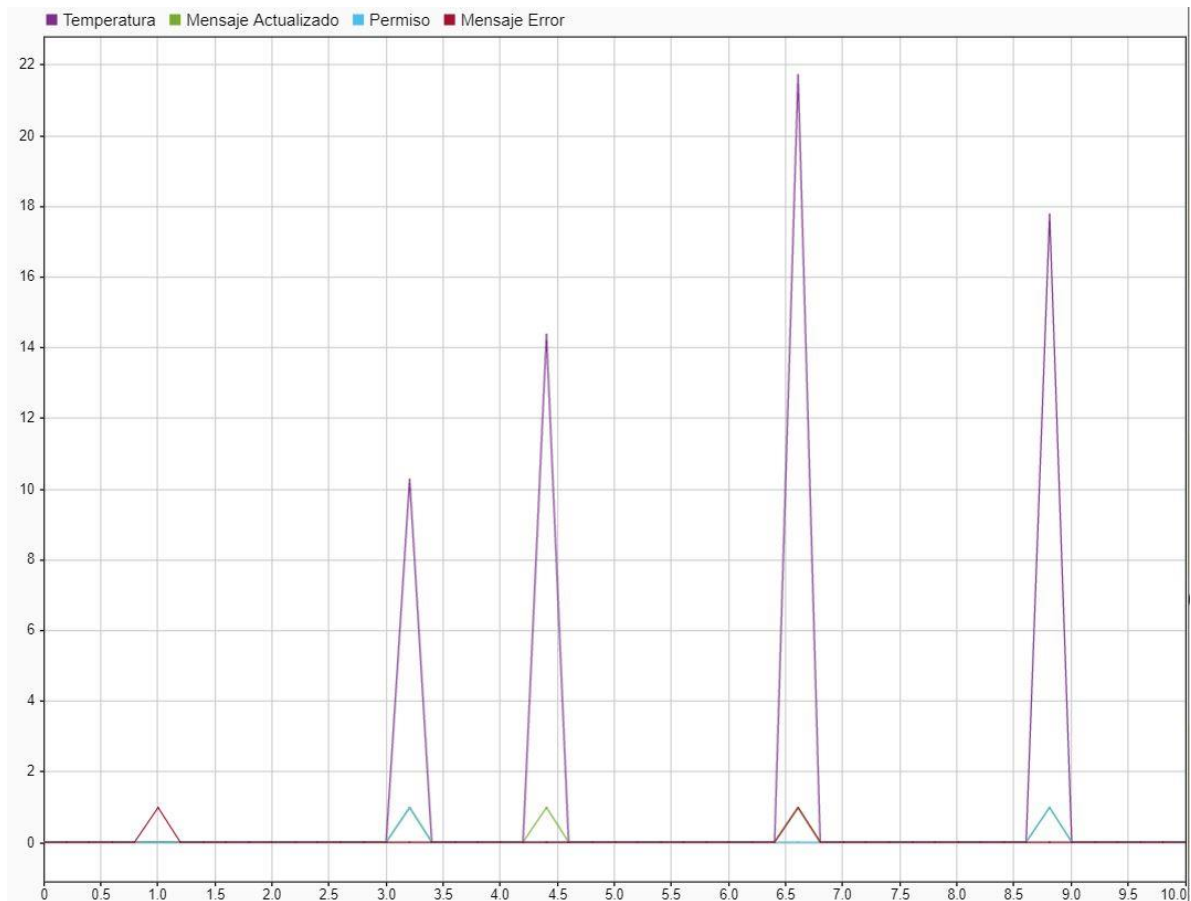


Ilustración 51: Resultado permiso nivel usuario y conexión parte meteorológico

Observamos en primer lugar un mensaje de error y por lo tanto no damos permiso a la actualización, ya que el nivel de usuario es de 3 con esta funcionalidad de nivel 2. Posteriormente tenemos nivel de usuario 1, con lo que si podrá tener acceso a actualizar parte meteorológico y a la vez los datos de temperatura y mensaje de parte actualizado.

El siguiente caso nadie pide actualización, pero pasa media hora, por lo tanto, también actualizaremos el parte. Más tarde, tendremos a un nivel de usuario 3 intentando actualizar, como no tiene permiso le sale un mensaje de error, pero justo lo ha pedido a la vez que ha pasado un día y por lo tanto actualizará por este requisito.

Finalmente tenemos un nivel de usuario 2 pidiendo actualizar, se le dará permiso y por lo tanto recibiremos los datos, pero no ocurrirá por la situación de que haya pasado una hora más al anterior test.

Step	Transition	Next Step
<b>Run</b> <pre>%% Initialize data outputs. nivelusuario = 0; funcionalidad = 0; hora = 0; minutos = 0;</pre>	1. <b>after(1,sec)</b>	step_1 ▼
<b>step_1</b> <pre>nivelusuario = 3; funcionalidad = 2; hora = 0; minutos = 0;</pre>	1. <b>true</b>	step_2 ▼
<b>step_2</b> <pre>nivelusuario = 0; funcionalidad = 0; hora = 0; minutos = 0;</pre>	1. <b>after(2,sec)</b>	step_3 ▼
<b>step_3</b> <pre>nivelusuario = 1; funcionalidad = 2; hora = 0; minutos = 0;</pre>	1. <b>true</b>	step_4 ▼
<b>step_4</b> <pre>nivelusuario = 0; funcionalidad = 0; hora = 0; minutos = 0;</pre>	1. <b>after(1,sec)</b>	step_5 ▼
<b>step_5</b> <pre>nivelusuario = 0; funcionalidad = 0; hora = 0; minutos = 30;</pre>	1. <b>true</b>	step_6 ▼

Ilustración 52: Banco de pruebas permiso nivel usuario y conexión parte meteorológico 1

<b>step_6</b> <pre>nivelusuario = 0; funcionalidad = 0; hora = 0; minutos = 0;</pre>	1. <b>after(2,sec)</b>	step_7 ▼
<b>step_7</b> <pre>nivelusuario = 3; funcionalidad = 2; hora = 24; minutos = 0;</pre>	1. <b>true</b>	step_8 ▼
<b>step_8</b> <pre>nivelusuario = 0; funcionalidad = 0; hora = 0; minutos = 0;</pre>	1. <b>after(2,sec)</b>	step_9 ▼
<b>step_9</b> <pre>nivelusuario = 2; funcionalidad = 2; hora = 0; minutos = 25;</pre>	1. <b>true</b>	step_10 ▼
<b>step_10</b> <pre>nivelusuario = 0; funcionalidad = 0; hora = 0; minutos = 0;</pre>		

Ilustración 53: Banco de pruebas permiso nivel usuario y conexión parte meteorológico 2

## Conclusiones

A lo largo de este documento, se han ido desarrollando diferentes pruebas a pequeñas implementaciones realizadas para cada una de las partes de nuestro sistema, siendo algunas, las más críticas para el correcto funcionamiento del sistema global.

Prácticamente todos los requisitos a los que se han realizado test se han podido verificar de forma adecuada. Aún así se ha podido ver como se han dado casos en los que se han establecido requisitos demasiados exigentes durante la redacción de estos, pero que, durante la elaboración de los test, no se han podido verificar de forma adecuada por cómo teníamos planteado nuestro sistema. Esto nos lleva a darnos cuenta de la gran interrelación entre estas dos partes de la elaboración de un proyecto de software, y cómo ambos han de estar en continua evolución. En nuestro caso particular, se ha visto cómo algunos requisitos con cierta exigencia realmente no eran tan críticos como se esperarían en un principio y sus exigencias se podrían haber disminuido, ya que el sistema seguiría funcionando de manera correcta, y de esta manera habiendo solventado todos los test de forma positiva.

Por concluir, ha sido una nueva forma de trabajar y elaborar software a la que nunca nos habíamos enfrentado, y que realmente vemos como la parte de desarrollo de pruebas automatizadas a nuestro software como una forma sencilla de tener siempre verificado que nuestro desarrollo cumple los requisitos y las exigencias que se requieren para el correcto funcionamiento de este. Es por ello por lo que vemos adecuado su implementación para próximos desarrollos en nuestra vida profesional.

## Apéndices

### Apéndice A: Glosario

<b>RFS</b>	<b>Requisito Funcional de Sistema</b>
<b>RFU</b>	<b>Requisito Funcional de Usuario</b>
<b>RNF</b>	<b>Requisito No Funcional</b>
<b>RNFD</b>	<b>Requisito No Funcional Desempeño</b>
<b>RNFS</b>	<b>Requisito No Funcional Seguridad</b>

<b>RNFME</b>	<b>Requisito No Funcional Manejo de Errores</b>
<b>RNFU</b>	<b>Requisito No Funcional de Usabilidad</b>
<b>RNFDI</b>	<b>Requisito No Funcional de Disponibilidad</b>
<b>RF</b>	<b>Requisito Funcional</b>
<b>RNFN</b>	<b>Requisito No Funcional de Normativa</b>