Trabajo Fin de Máster

Máster Universitario en Ingeniería Electrónica, Robótica y Automática

Diseño e implementación de interfaz de comunicación entre plataformas MATLAB y Home I/O

Autor: Javier Jiménez Sicardo

Tutor: José Mª Maestre Torreblanca

Dpto. Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2021

Trabajo Fin de Máster
Máster Universitario en Ingeniería Electrónica, Robótica y Automática

Diseño e implementación de interfaz de comunicación entre plataformas MATLAB y Home I/O

Autor:
Javier Jiménez Sicardo

Tutor:
José María Maestre Torreblanca
Catedrático de Universidad

Dpto. de Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2021

Trabajo Fin de Máster: Diseño e implementación de interfaz de comunicación entre plataformas MATLAB y Home I/O

Autor: Javier Jiménez Sicardo

Tutor: José Mª Maestre Torreblanca

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2021

El Secretario del Tribunal

A mi familia

A mis maestros

Agradecimientos

A quienes hayan ayudado a facilitarme la conclusión de
esta carrera universitaria.
No tiene por qué ser profesor, familiar o amigo,
ni tan siquiera encontrarse en el ámbito académico.
Un mero apoyo en momentos de flaqueza es suficiente.

Dése por aludido quien así lo crea. Muchas gracias.

Javier Jiménez Sicardo

Máster Universitaario en Ingeniería Electrónica, Robótica y Automática

Sevilla, 2021

Resumen

En la investigación, a veces hay cierta preferencia a utilizar algunos programas más que otros. En el campo de la domótica hay varios programas que suelen utilizarse para simular viviendas domotizadas, y uno de ellos es *Home I/O*. El propósito de este trabajo dar los pasos previos a una investigación sobre consumo energético en un hogar con prestaciones domóticas, y explorar técnicas que pretendan mejorar la eficiencia energética (y con ello el coste de la factura eléctrica) de sus habitantes. No obstante, para ello es necesario desarrollar una buena interfaz de comunicaciones primero; y de eso es de lo que trata este trabajo.

Dado el volumen de investigación que se realiza últimamente con Home I/O en este trabajo se dirigen los esfuerzos del trabajo a crear una interfaz de comunicaciones hacia MATLAB, que sea rápida y lo suficientemente sencilla de utilizar, de modo que se simplifique todo el trabajo posterior que realizar a la hora de obtener resultados de investigación posteriores.

Abstract

When researching, there are always leaning when choosing to use some simulation software over others. Regarding home automation, there is just a handful software viable to run a whole house simulation, one of them being *Home I/O*. The main purpose of this Thesis is starting to perform small steps towards a goal, which is researching on power consumption at home and explore new techniques, allowing the inhabitants to become more energy efficient and save on the electricity bill costs. However, there is still a gap for accomplishing this task, which is that there is currently no interfaces between *MATLAB* and the *Home I/O* software.

Due to the high volume on the research that is being conducted in *Home I/O*, the main effort of this Thesis goes towards the development of a communications interface between this virtual home software and MATLAB, fast and easy to use enough, so that all later work in obtaining other results would, hopefully, get streamlined.

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1 SITUACIÓN CONTEXTUAL Y JUSTIFICACIÓN

Bien es sabida la amplia y variada producción científica en la Escuela Técnica Superior de Ingeniería (*ETSI*) de la Universidad de Sevilla (*US*). Entre los campos de investigación del Departamento de Ingeniería de Sistemas y Automática (*DISA*) se encuentra la Domótica, área que, en una definición informal, trata de aplicar técnicas de automatización y control de sistemas a dispositivos físicamente instalados en una vivienda, persiguiendo la obtención una mejora en el bienestar de los habitantes de la referida vivienda (que se convierten en los operadores del sistema automatizado) en términos de comodidad, confort, seguridad, ahorro energético (y económico) y otros tantos parámetros que se presten a consideración, frente a la situación no controlada, en que la vivienda se encontraría simplemente cableada manteniendo una instalación eléctrica tradicional y sin capacidad de actuación de forma automatizada sobre los dispositivos más importantes instaladas en ésta.

Figura 1.1 Imagen libre de derechos en referencia a la domótica (Gerd Altmann en Pixabay)

De estos mencionados beneficios en cuanto a bienestar que resultan de la aplicación de la domótica en una vivienda (o de la inmótica, hablando de inmuebles en caso de que no sean estrictamente una vivienda unifamiliar) también se tiene amplia concienciación y conocimiento de forma general en la sociedad. Si el hogar o el edificio está lo suficientemente bien equipado de sensores y actuadores, hoy día no resulta ningún tipo de ficción (y es decir más, ya se encuentra totalmente normalizado e integrado en la sociedad) permitirse decir, de nuevo en términos informales, frases desde "*no tengo que levantarme para encender y apagar las*

luces (o subir o bajar las persianas cuando amanece o anochece) porque lo hago desde mi móvil", hasta "*la vivienda me avisa si detecta algún tipo de intrusión, se va la luz, detecta fuego o inundaciones*", pasando por "*me puedo despreocupar de encender y apagar la calefacción porque la vivienda sabe dónde estoy y qué temperaturas prefiero*" y, naturalmente, "*la gestión del consumo energético la hace mi casa, no yo*".

También es bien conocida la situación actual del papel de la vivienda en la sociedad actual, como sistema habilitado a través del entramado paradigmático de la Sociedad de la Información y la Industria Conectada 4.0 (también conocido como *Industria 4.0*) como otra cosa más del *Internet de las Cosas*. Todo esto debe estar suficientemente contextualizado en un mundo que empuja cada vez más y más a conectar hasta el último dispositivo que sea susceptible de tener cierta inteligencia, a veces con mayor o menor acierto, como bien apuntaron algunas empresas como Cisco (dedicada fundamentalmente a redes y dispositivos de telecomunicaciones), que ya acuñó a principios de la década 2010-2020 el concepto de *Internet of Everything* [1] (Internet de Todo) haciendo referencia (más allá de un posible interés corporativo) a que no solo sería la vivienda o la industria lo que estuviera conectado en un futuro, sino hasta las carreteras y también la última farola o pieza de mobiliario urbano de las sociedades modernas. Aunque todavía no se ha materializado este futuro, es una predicción que parece estar cumpliéndose, y desde una visión actual sí tiene buenas posibilidades de que así será en algún momento.

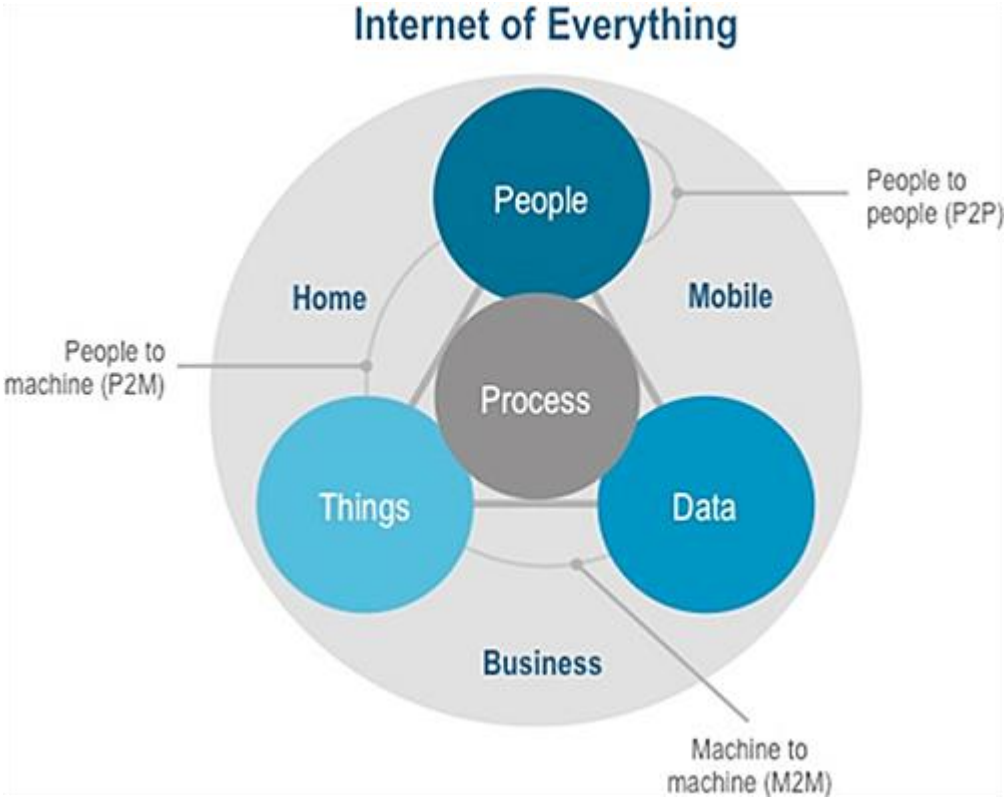


Figura 1.2 Inclusión de personas, datos y cosas en el Internet de Todo, según *D. Evans* (Cisco, 2012) [2]

En virtud de la novedad de la situación, podrían hacerse referencias a que el Internet de Todo y los esfuerzos por la progresiva digitalización de cualquier cosa que se ocurra, entre los gemelos digitales y demás, podría comenzar a hablarse del metaverso, que ya escapa la realidad física para pasar a una realidad digital en un ecosistema virtual [3]. No obstante, por muy interesantes y por muy amplio que se pueda hablar al respecto de estas consideraciones, este documento no trata del metaverso.

Este trabajo tampoco trata del Internet de Todo aunque, como se ha podido comprobar, sí que se podría abordar esta temática, haciendo tal vez un inciso que cubriera los problemas o preocupaciones que sí surgen cuando los dispositivos del Internet de las Cosas se instalan y que pueden traer a una vivienda dotada de suficiente domótica (o más en general, a un edificio dotado de suficiente inmótica), principalmente en lo que concierne a seguridad y evitación de accesos indeseados [4], más aún cuando en el todavía prometedor

mundillo del Internet de las Cosas [5], y dentro de él en la Domótica [6] [7], sigue habiendo pugna por la imposición de un protocolo estándar común, lo cual es lógico porque cada tecnología responde a diferentes necesidades. Esta no existencia de un estándar común acaba dificultando la interoperabilidad de los dispositivos o incluso causando que los fabricantes tengan que implementar más de un protocolo de comunicaciones para permitir una interoperabilidad, con los costes esto comporta, tanto en su diseño como en el uso de recursos, llegando a impactar en el coste hacia el usuario. Algunos de estos estándares ya competían antes simplemente de forma circunstancial dada la gran penetración que ya tenían al ser ya previamente consideradas (en algún caso incluso por antonomasia) de pleno derecho en las redes de área doméstica y personal; y han dirigido sus esfuerzos también para desarrollarse en su vertiente tanto para IoT o domótica. Varios de los logos de estas tecnologías que pueden ser utilizadas en domótica (todas ellas mencionadas en [7]) se muestran en las figuras 1.3 a 1.7 (sin ningún orden en particular). Estos logos son propiedad (en caso de tenerla) de sus respectivas marcas.

Figura 1.3Algunas tecnologías que se pueden utilizar en domótica

Figura 1.4 Más tecnologías que se pueden utilizar en domótica

Figura 1.5 Más tecnologías que se pueden utilizar en domótica

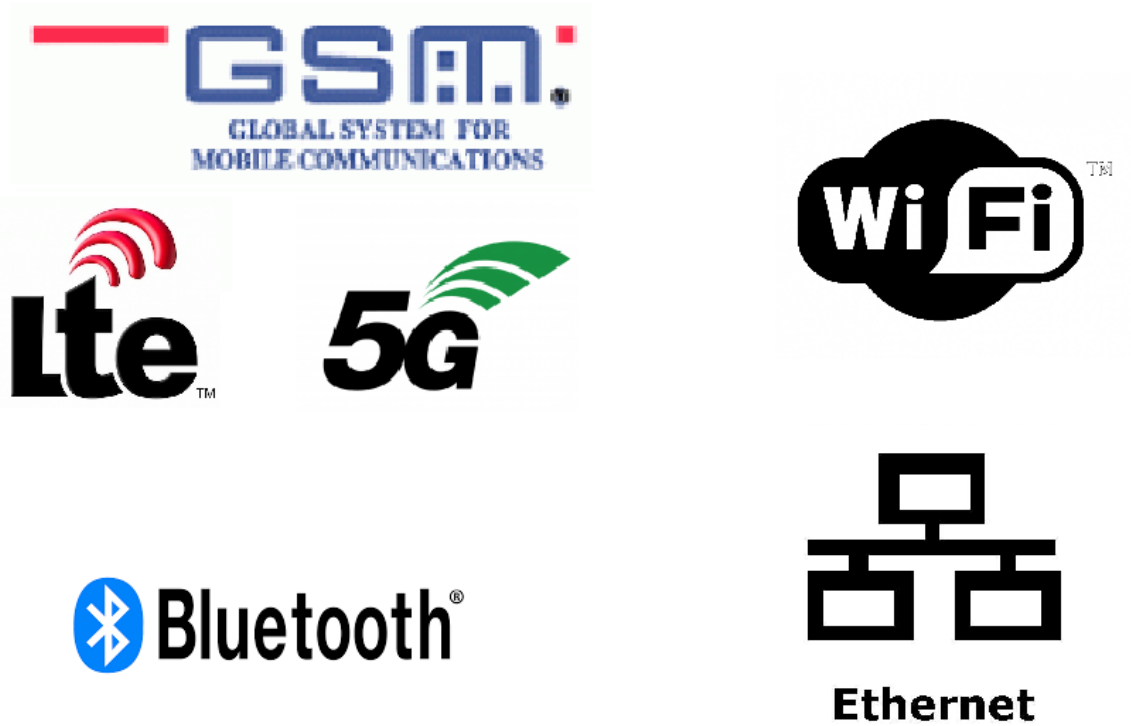


Figura 1.6 Más tecnologías que se pueden utilizar en domótica

No obstante de lo anterior, y aunque se ha querido dar los logos de las tecnologías domóticas e IoT para causar cierta familiaridad y provocar una asociación visual entre el lector y la correspondiente tecnología, además de facilitar una puesta en contexto, el desarrollo de este documento tampoco va a hacer uso de las mismas. En cambio, para evitar algunos problemas que podrían dar estas tecnologías en su uso real y para facilitar los primeros pasos (o en ocasiones pasos previos) en las líneas de investigación del Departamento, se utilizará un simulador de vivienda llamado *Home I/O*, al cual se hará referencia con posterioridad.

Habiendo puesto este trabajo en contexto con los párrafos anteriores, justificarlo resulta sencillo. Retomando la última de las afirmaciones dadas de manera informal al principio de este texto ("*la gestión del consumo energético la hace mi casa, no yo*"). Si la vivienda gestiona el consumo energético, de forma indirecta también está gestionando el consumo económico de los habitantes en cuanto al término de energía (KWh) de la factura de la luz, lo cual puede resultar de especial relevancia e interés si se presta atención a la tendencia creciente (según la fecha de este TFM) que tiene el precio del suministro eléctrico en los últimos meses, según la web del Operador del Mercado Ibérico de Energía (*OMIE*), en su polo español [8], tal como indica la figura 1.7.

Figura 1.7 Evolución diaria de los precios de la electricidad en el sistema eléctrico

Más allá de posibles causas y responsabilidades que apuntan a problemas exógenos al sistema eléctrico ibérico y apuntan a que esta subida de precios se debe a una acumulación de factores tales como la bajada en reservas de gas natural, lo que ha provocado una subida de su precio; y al control de precios de este recurso al que puede estar sometiendo Rusia a la Unión Europea, causando esto a su vez un repunte en los precios del carbón y provocando cierta inestabilidad en los mercados energéticos europeos [9] [10] [11]. Una explicación más detallada no compete a este texto.

Entonces, el trabajo que se pretende realizar y sobre el que se quieren dar los primeros pasos sigue estas líneas: la de obtener datos de prueba de este simulador de vivienda, *Home I/O*, aprovechando sus capacidades y darles un tratamiento analítico que permita obtener información de su comportamiento térmico y consumo energético, como puede ser obtener un modelo, ensayarlo y aplicarle algún tipo de control para comprobar si se obtiene alguna mejora en cuanto a la gestión energética de la vivienda.

Como se hace necesario obtener una cantidad grande de información del simulador, se decide comenzar creando una interfaz de comunicación entre la vivienda simulada desde *Home I/O* y *MATLAB* como paso previo y objetivo principal de este trabajo, aprovechando para poder volcar todo el resto de la información de la vivienda en caso de que fuera necesario o se requiera en el futuro para cualquier otro tipo de aplicaciones, aunque el enfoque principal de este trabajo se dará al comportamiento térmico y energético de la vivienda porque, como ya se verá, son los calefactores los dispositivos que más energía consumen y facilitarán el trabajo en este respecto.

2 TECNOLOGÍAS EMPLEADAS

Las tecnologías empleadas en la realización de este trabajo son principalmente dos: MATLAB como programa de adquisición y tratamiento de datos; y Home I/O, como simulador de vivienda.

2.1. MATLAB

MATLAB [12] es un software informático es muy conocido en el ámbito de las matemáticas, las ciencias y las ingenierías, tan conocido y de uso tan extendido en el ámbito académico que apenas precisa presentación. Se trata de un software privativo y de pago de cálculo numérico que incluye una *IDE* (interfaz de desarrollo integrada, del inglés *Integrated Development Environment*) y es a su vez su propio lenguaje de programación, comúnmente conocido porque los archivos programados en *MATLAB* llevan la extension *m* (aunque puede utilizar otras extensiones de archivo para otras funcionalidades que tiene). Lleva en desarrollo continuo y activo desde 1984 hasta la actualidad. En el presente, la empresa encargada de su desarrollo y explotación económica desde el año 1994 es *The Mathworks, Inc.*

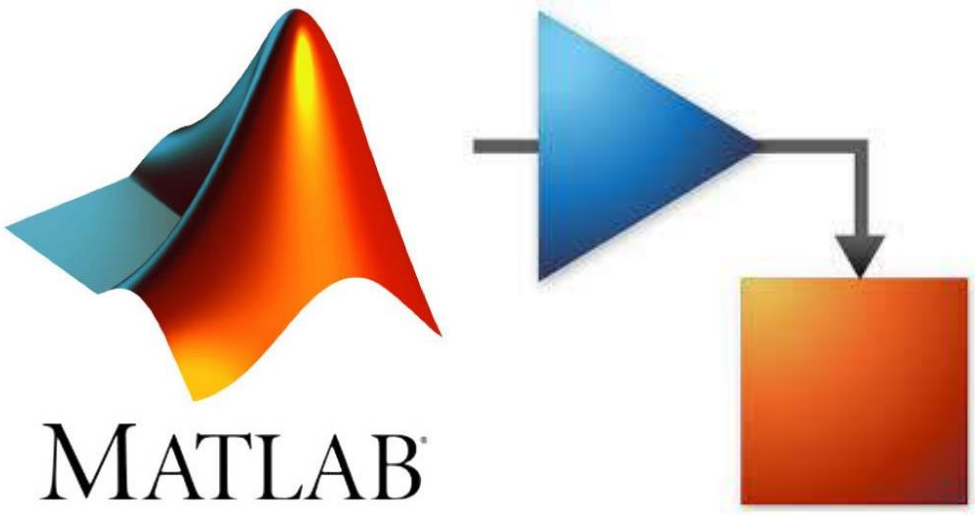


Figura 2.1 Logotipos de *MATLAB*®, a la izquierda; y *Simulink*®, a la derecha (*propiedad de MathWorks*®)

Además de su *IDE* y ser su propio lenguaje de programación, la potencia de *MATLAB* y la utilidad que tiene en los diferentes campos de la ciencia se suelen dejar fuera de toda duda debido a que no es una aplicación que trabaje totalmente en solitario, sino que sus funcionalidades han ido siendo ampliadas y completadas continuamente (construyendo sobre la base, que es este propio software *MATLAB*) e incluye como complemento básico (de serie con *MATLAB*) un lenguaje visual basado en bloques como es *Simulink*. El resto de mejoras y ampliaciones sobre estos softwares básicos, de gran cantidad y variedad, suelen ser llamados *toolboxes* (cajas de herramientas) y suelen estar diseñados para aplicaciones específicas y de carácter más de tipo especialista (mecánica, electronica de potencia, bioingeniería, análisis econónico, identificación de sistemas, interfaz con otros lenguajes de programación, entre otros), que requieren un pago adicional para estar incluidos en la licencia del usuario. Afortunadamente, la Universidad de Sevilla ofrece a sus alumnus una licencia completa de *MATLAB*.

A pesar de que hay alternativas a *MATLAB* [13] (algunas gratuitas y libres como *GNU Octave* y *Scilab*, o incluso programando en *Python* con *SciPy* y *NumPy*), se ha preferido utilizar este software por la versatilidad y robustez que ofrece, además de su renombre y sus capacidades.

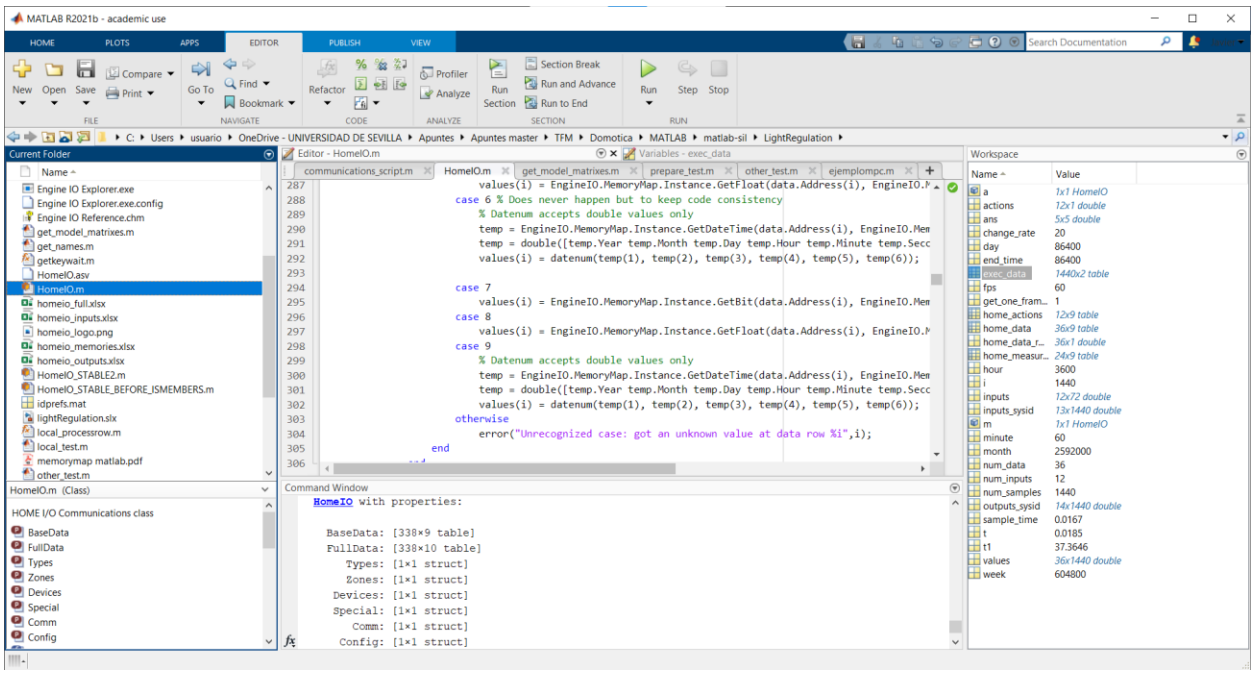


Figura 2.2 Ventana del *IDE* de *MATLAB* con la clase de comunicación con *Home I/O* a la vista

En términos generales, el propósito del trabajo con *MATLAB* es la comunicación con *Home I/O* a través del SDK (Kit de Desarrollo de Software, del inglés *Software Development Kit*) que este simulador de vivienda ofrece, haciendo llamadas a los datos que *Home I/O* permite intercambiar. De la obtención de estos datos, se procederá en un trabajo posterior a la identificación de la casa completa y al control de sus sistemas de calefactores para aplicaciones de eficiencia energética. No obstante, como ya se mencionó en el capítulo 1, se buscará la obtención e intercambio de datos de la casa simulada, por completo, y ,únicamente en caso de ser necesario, se prestará atención adicional a los parámetros que influyen en los calefactores o son resultado directo de su control (como las temperaturas de cada sala, de forma totalmente obvia).

Debido al gran número de características que ofrece *MATLAB*, se irán enlazando a cuestiones en la documentación, ayudas y referencias del software que hayan sido utilizadas en el desarrollo antes que inundar este documento de páginas y páginas sobre características, antes de hacerse necesarias, lo cual puede ser causante de confusión al lector.

2.2. Home I/O

Home I/O [14], tal como se ha avanzado previamente en varias ocasiones, es un software de simulación de vivienda, que ha sido desarrollado por la compañía portuguesa RealGames [15] (empresa que ha desarrollado otros simuladores 3D que también se utilizan en la *ETSI*, como el *ITS PLC* y el *Factory I/O*, para entrenamiento con autómatas programables) en colaboración con *CReSTIC* [16], el Centro de Investigación en Ciencias y Tecnologías de la Información de la Universidad de Reims Champagne-Ardenne, y galardonado con un reconocimiento de interés pedagógico por el Ministerio de Educación francés [14]. Este software es también privativo, con un modelo de acceso por pago único, aunque también ofrece una versión de prueba gratuita e ilimitada durante 30 días. Como herramientas de apoyo se tiene una documentación que también sirve como manual de instrucciones tanto para su uso normal como para quien pretenda utilizar su SDK [17], disponible tanto en inglés como en francés.

Tal como se ha mencionado, este software de simulación permite simular una vivienda domotizada en 3D ofreciendo una interfaz hacia el usuario similar a un videojuego, lo que permite navegar por la casa con gran familiaridad y sencillez, permitiendo incluso la navegación con un mando de consola en caso de ser necesario [18].

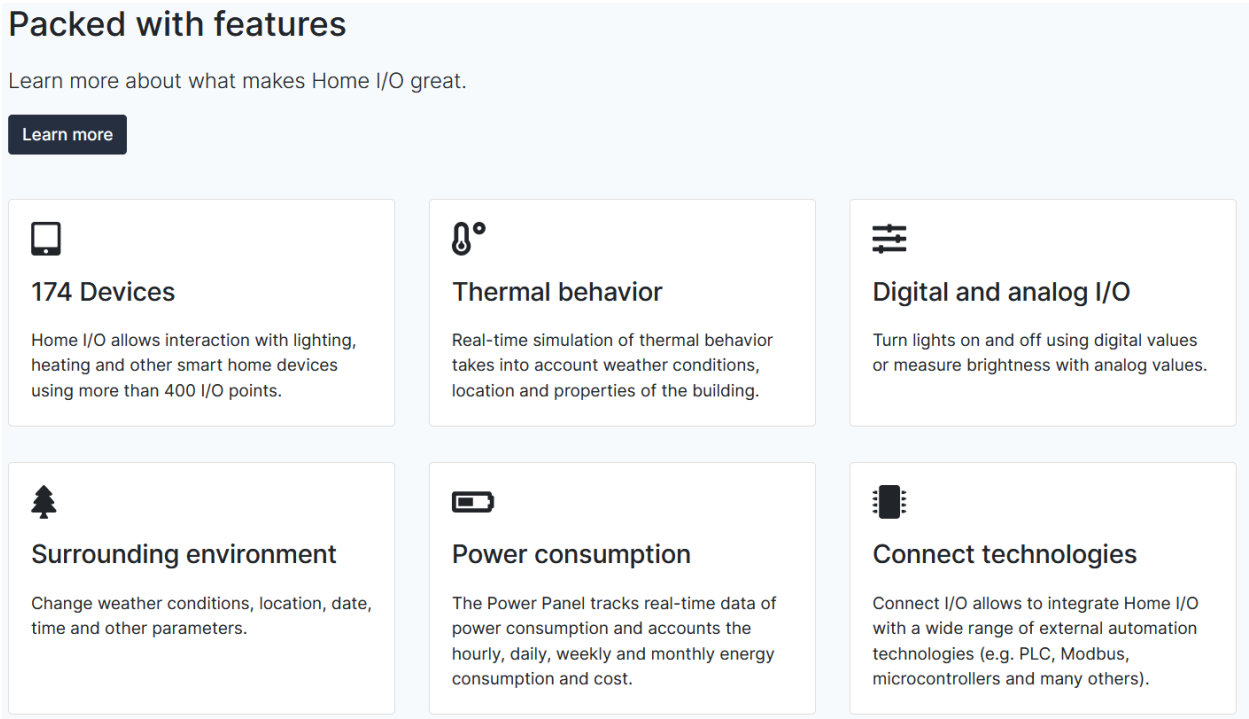


Figura 2.3 Captura de la web de *Home I/O* que indica las características que ofrece

La disposición de la vivienda que ofrece el simulador *Home I/O* (una casa de campo, tipo chalé de dos plantas, de corte moderno y con ciertos lujos, situado a las afueras de la civilización en una ubicación que parece ser de montaña boscosa, con un lago en las proximidades de la casa) es única y no se puede cambiar, aunque sí que permite al usuario cambiar y ajustar de forma inmediata y en línea varios parámetros de la simulación, como son la velocidad de simulación (hasta 1:5000 s), el rango de temperaturas exteriores, la humedad, el viento y la nubosidad; además de la fecha y hora y la localización en latitud y longitud, que permiten posicionar de forma realista el Sol y la Luna y calcular la incidencia (solar) sobre la vivienda simulada. *Home I/O* no permite simular otros fenómenos climatológicos más allá de días soleados o soleados nubosos.

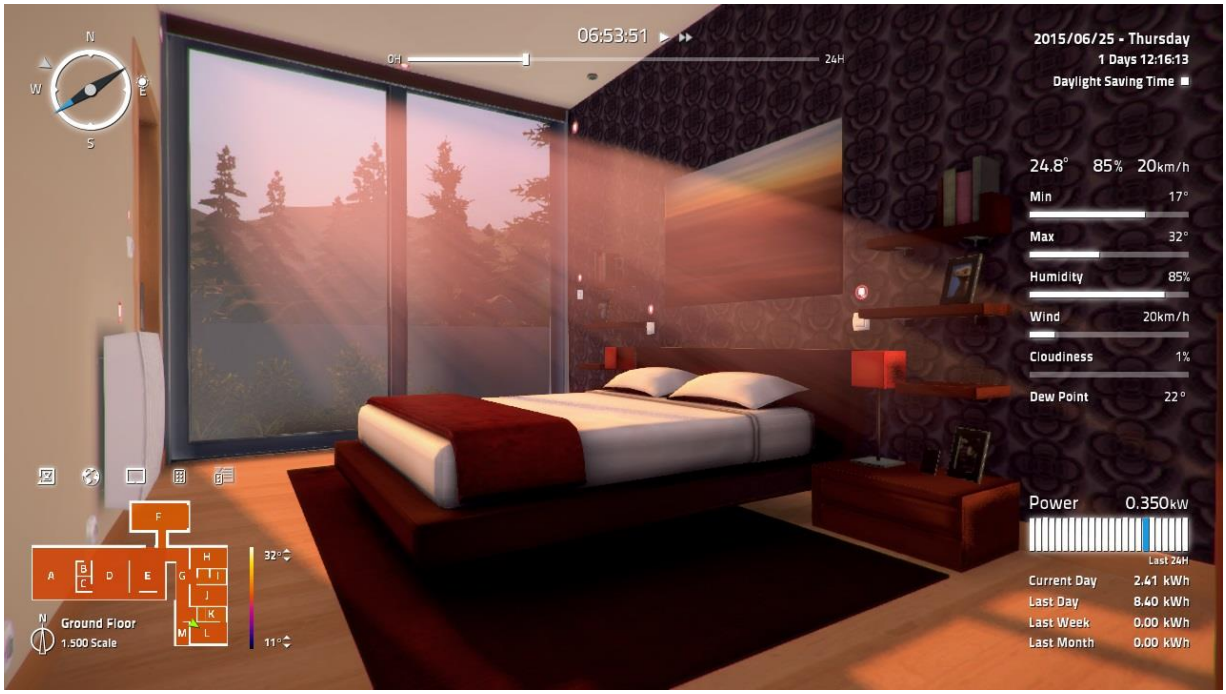


Figura 2.4 Imagen de muestra de *Home I/O*, tomada de su propia web

2.2.1. Distribución de la vivienda

La distribución de estancias de la vivienda es como en la tabla que sigue (Tabla 2.1), en la que la columna "Zona" hace referencia al nombre interno de la estancia para *Home I/O* (manteniendo la nomenclatura que se da en la documentación en la web de *Home I/O*) y la columna "Habitación" se refiere a la zona que tiene sentido lógico para las personas que supuestamente fueran a habitar la casa, traducido al español:

Zona	Habitación	Planta
A	Sala de estar	Baja
B	Aseo de invitados	Baja
C	Despensa	Baja
D	Cocina	Baja
E	Recibidor	Baja
F	Garaje	Baja
G	Pasillo	Baja
H	Dormitorio infantil	Baja
I	Cuarto de baño	Baja
J	Dormitorio simple	Baja
K	Cuarto de baño privado	Baja
L	Dormitorio de pareja	Baja
M	Lavadero	Baja
N	Despacho	1ª
O	Exterior	(ninguna)

Tabla 2.1 Relación de estancias de la vivienda de *Home I/O*

Cada estancia de esta vivienda dispone de un número variable de sensores y actuadores dispuestos en ella que permiten interactuar sobre la misma (según el propósito de la estancia, supuesta equivalente a la actividad que se haría en ellas en caso de estar habitando la casa en la realidad); y cuenta y ejecuta en el fondo varios modelos matemáticos para simular el comportamiento térmico de la vivienda y las condiciones ambientales de la misma [19], así como estimar el consumo energético agregado que se tiene por el uso de iluminación, por calefacción y por el consumo de otros dispositivos instalados.

Para consulta de la distribución física de los espacios construidos virtualmente en la vivienda de *Home I/O*, en la sección siguiente se muestra un mapa sinóptico que incluye los calefactores y termostatos dispuestos a través de las diferentes estancias.

2.2.2 Tipos de dispositivos

La vivienda se encuentra totalmente modelada, pero solo son funcionales (e interactuables) un rango limitado de dispositivos, lo que puede responder a que los desarrolladores de *Home I/O* han dirigido sus esfuerzos a hacer una simulación funcional con un conjunto pequeño de categorías de aquellos aparatos más típicos y susceptibles de ser controlados a través de domótica, en lugar de dedicarse a implementar funcionalidades que no tengan tanto impacto o que no consideraran importantes.

Por tanto, el consumo de electrodomésticos tales como la televisión, la cocina, la nevera, la lavadora, el termo eléctrico de agua; o de otra electrónica como los ordenadores personales distribuidos por la vivienda, la guitarra eléctrica de la habitación individual o las videoconsolas que hacen símil a unas Nintendo 3DS, entre otros aparatos eléctricos que se pueden ver en las diferentes estancias (así como de las tomas de corriente dispuestas en la vivienda), no se tienen en consideración, por lo que no se puede interactuar con ellos. De igual manera, tampoco es posible hacer uso de cualquier aparato relacionado con el suministro de agua en esta vivienda simulada: no se ha previsto ningún tipo de interactuación con grifos ni ningún otro aparato similar.

Figura 2.5 Ejemplo de dispositivos que no ofrecen posibilidad alguna de interactuar, en la sala H

Por otra parte, aunque haya dispositivos instalados en la casa que tienen una apariencia diferente, su funcionamiento es idéntico siempre y cuando éstos sean de la misma categoría (sí pueden tener un consumo energético diferente, como se verá con posterioridad).

Figura 2.6 Calefactor (*radiador*) de sala visible en la mayoría de estancias, a la izquierda; y calefactor de toallas visible únicamente en los cuartos de baño (el indicador rojo señala que se encuentran encendidos)

Aunque la documentación web trata en detalle cada uno de los dispositivos en el apartado *devices*, con varias páginas, aquí se prefiere hacer una tabla resumen (traducida al español), similar a la que aparece en la página *Connecting to External Technologies* [20], porque se tratará con posterioridad una tabla completa con todos los dispositivos de la casa que se mantendrá en el idioma original (inglés) dado que esa será la lengua que se utilizará en la redacción del código.

Cuando en las siguientes entradas se mencione una "situación normal", se refiere al estado por defecto de la vivienda (con todos los dispositivos en modo cableado). Se profundizará en más detalles al respecto con posterioridad este mismo capítulo.

2.2.2.1 Interruptores

Se trata de simples interruptores de encendido y apagado.

Etiqueta	Tipo E/S	Tipo	Descripción
(Ninguna)	Entrada	Bit	Verdadero si se pulsa

Tabla 2.2 Valores comunes a los interruptores

Es un monoestable con dos posiciones, con un estado estable de apagado.

2.2.2.2 Atenuadores de luz

Son visualmente parecidos a los interruptores, pero permiten aumentar o disminuir la intensidad lumínica de la luz que controlan.

Etiqueta	Tipo E/S	Tipo	Descripción
Arriba	Entrada	Bit	Verdadero si se pulsa arriba
Abajo	Entrada	Bit	Verdadero si se pulsa abajo

Tabla 2.3 Valores comunes a los atenuadores de luz

Ambas entradas son monoestables de dos posiciones, con estado estable de apagado.

2.2.2.3 Pulsadores arriba y abajo

Son los interruptores que controlan los motores de las persianas instaladas en la casa.

Etiqueta	Tipo E/S	Tipo	Descripción
Arriba	Entrada	Bit	Verdadero si se pulsa arriba
Abajo	Entrada	Bit	Verdadero si se pulsa abajo

Tabla 2.4 Valores comunes a los pulsadores arriba y abajo

2.2.2.4 Sensores de movimiento

Son sensores que se activan si detectan movimiento en su rango de detección. Su estado activo se señaliza con un indicador rojo.

Etiqueta	Tipo E/S	Tipo	Descripción
(Ninguna)	Entrada	Bit	Verdadero si detecta movimiento

Tabla 2.5 Valores comunes a los sensores de movimiento

2.2.2.5 Sensores de humo

Son sensores que se activan si detectan humo (se puede utilizar una bengala si se pulsa la tecla 1 en las configuraciones por defecto). Su estado activo se señaliza con un indicador rojo, emitiendo un pitido mientras continúe detectando humo.

Etiqueta	Tipo E/S	Tipo	Descripción
(Ninguna)	Entrada	Bit	Verdadero si detecta humo

Tabla 2.6 Valores comunes a los sensores de humo

2.2.2.6 Sensores de luminosidad

Son sensores que se activan si detectan humo (se puede utilizar una bengala si se pulsa la tecla 1 en las configuraciones por defecto). Su estado activo no se señaliza de forma visible.

Etiqueta	Tipo E/S	Tipo	Descripción
(Ninguna)	Entrada	Bit	Verdadero si detecta poca luminosidad (menos de 3V con histéresis de ± 1V)
Analógico	Entrada	Flotante	Valor de la intensidad lumínica [0-10 V]

Tabla 2.7 Valores comunes a los sensores de luminosidad

El cambio de bit se señaliza únicamente en caso de que se supere la histéresis indicada de ± 1V

2.2.2.7 Sensores de puerta y ventana

Son sensores que se activan al abrir y cerrar las puertas o ventanas. Su estado activo se señaliza con un indicador, siendo el indicador de color rojo al cerrar la puerta o ventana; y de color verde cuando al abrir.

Etiqueta	Tipo E/S	Tipo	Descripción
(Ninguna)	Entrada	Bit	Verdadero si puerta/ventana cerrada

Tabla 2.8 Valores comunes a los sensores de puerta y ventana

2.2.2.8 Termostatos

Permiten ver la temperatura actual de la sala en la que se encuentren, y cambiar el punto de referencia para el control de temperatura integrado en *Home I/O*.

Etiqueta	Tipo E/S	Tipo	Descripción
Temperatura de la estancia	Entrada	Flotante	Temperatura de la estancia
Temperatura de referencia	Entrada	Flotante	Temperatura de referencia

Tabla 2.9 Valores comunes a los termostatos

Las temperaturas de referencia en los termostatos de Home I/O no se utilizarán al no poder modificarse los valores de tipo entradas (como se verá más adelante). En su lugar, las referencias se darán de forma externa.

2.2.2.9 Alarma centralizada

Es un sistema de seguridad centralizado (único en la vivienda). Se puede armar y desarmar (señalizado con iconos de candado cerrado y abierto, respectivamente) tanto a través de un mando de control remoto como manualmente a través de su panel numérico; y hace sonar la sirena de la vivienda en caso de detectarse intrusión con el sistema armado.

Etiqueta	Tipo E/S	Tipo	Descripción
Alarma armada	Entrada	Bit	Verdadero si la alarma se encuentra armada
Armar alarma	Salida	Bit	Arma el sistema de alarma en cuanto se activa el bit
Desarmar alarma	Salida	Bit	Desarma el sistema de alarma en cuanto se activa el bit

Tabla 2.10 Valores propios de la alarma centralizada

2.2.2.10 Luces

Es la iluminación de la vivienda en cada estancia. En una situación normal, se encienden o apagan debido al accionamiento de un interruptor o un atenuador de luz.

Etiqueta	Tipo E/S	Tipo	Descripción
(Ninguna)	Salida	Bit	Enciende o apaga las luces
Analógico	Salida	Flotante	Valor de la intensidad de las luces [0-10 V]

Tabla 2.11 Valores comunes a las luces

No se indica en la documentación, pero los valores flotantes que se den a las luces toman prioridad frente a los valores de bit. Es decir, el valor binario de las luces se ignora si hay un valor entre 0 y 10 para controlar las luces, llegando a producirse situaciones como que, en caso de apagar una luz (por la vía analógica) llegue a encenderse por completo quedar anteriormente su bit activo. Se harán consideraciones a este respecto más adelante.

2.2.2.11 Persianas

Son persianas que permiten controlar la incidencia de la luz del exterior en el hogar. En situaciones normales, se accionan a través de los pulsadores de arriba y abajo.

Etiqueta	Tipo E/S	Tipo	Descripción
Arriba	Salida	Bit	Sube la persiana
Abajo	Salida	Bit	Baja la persiana (domina si están activos el bit de arriba y abajo a la vez)
Grado de apertura	Entrada	Flotante	Grado de apertura de la persiana [0-10 V]. 0 si totalmente cerrada

Tabla 2.12 Valores comunes a las persianas

2.2.2.12 Sirena

Es el actuador (único en la vivienda) que, en situaciones normales, activaría la alarma en caso de detectar intrusión. Su estado activo se señaliza con sonido y con un indicador de color rojo.

Etiqueta	Tipo E/S	Tipo	Descripción
(Ninguna)	Salida	Bit	Enciende o apaga la sirena

Tabla 2.13 Valores propios de la sirena

2.2.2.13 Calefactores

Son los calefactores de la vivienda. En situaciones normales, son regulados a través de las temperaturas de referencia que se indiquen a los termostatos, con un control por relé (todo-nada) con 1ºC de histéresis.

Etiqueta	Tipo E/S	Tipo	Descripción
(Ninguna)	Salida	Bit	Enciende o apaga el calefactor
Analógico	Salida	Flotante	Valor de la potencia consumida por el calefactor [0-10 V]

Tabla 2.14 Valores comunes a los calefactores

No se indica en la documentación, pero los valores flotantes dados a los calefactores toman prioridad frente a los valores de bit. Es decir, el valor binario de los calefactores se ignora si hay un valor entre 0 y 10 para controlarlos, llegando a producirse situaciones como que, en caso de apagar calefactor (por la vía analógica) llegue a encenderse por completo al estar su bit activo. Se harán consideraciones a este respecto más adelante.

La página de documentación de *Home I/O* ofrece mapas de la vivienda indicando los dispositivos instalados por grupos [21], para no adjuntar seis veces el mismo mapa con información ligeramente diferente (para lo que está la fuente), y dado que se trabaja con especial interés en los calefactores de este hogar virtual, se adjunta únicamente el mapa con la localización de cada uno de ellos en los planos de la vivienda:

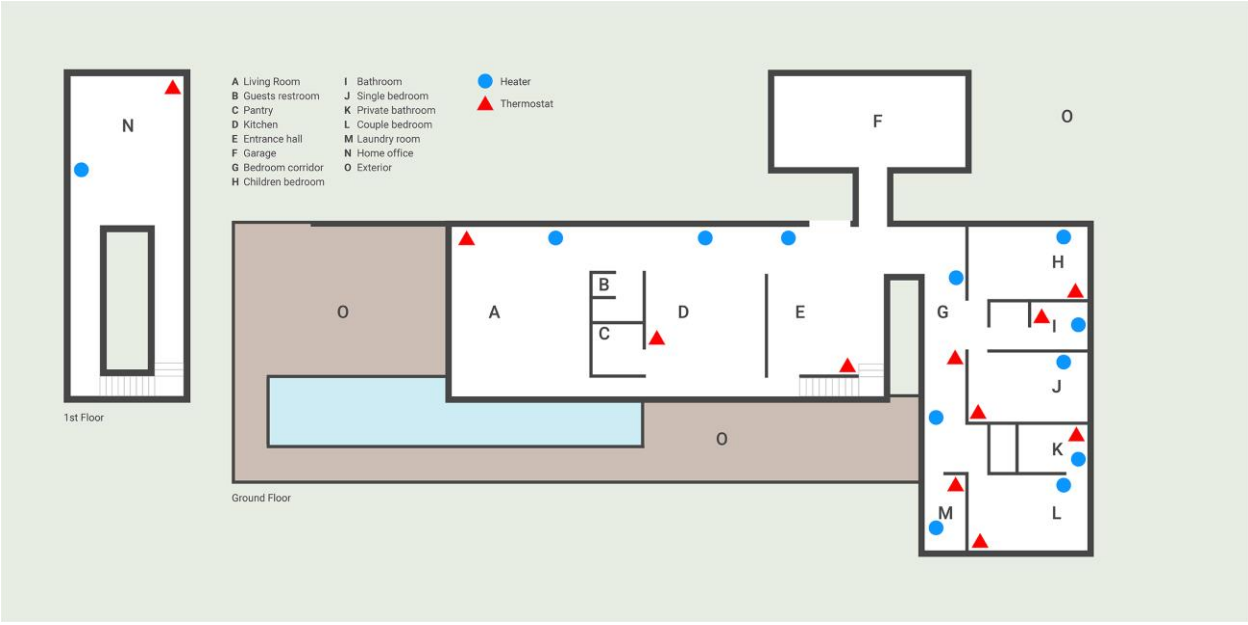


Figura 2.7 Mapa de la vivienda *Home I/O* con la disposición de todos los calefactores y termostatos

2.2.2.14 Puerta del garaje

Es una puerta de movimiento vertical, única en la vivienda, que permite la entrada y salida al garaje de la casa. En una situación normal se puede abrir a través del mando de control remoto o de un pulsador arriba y abajo dentro del garaje. Solo es posible saber si el motor de la puerta está activo, tanto para abrir como para cerrar, porque se señala con un indicador rojo. Los sensores (entradas) no dan señales visibles.

Etiqueta	Tipo E/S	Tipo	Descripción
Abrir	Salida	Bit	Sube la puerta del garaje (domina si están activos el bit de arriba y abajo a la vez)
Cerrar	Salida	Bit	Baja la puerta del garaje
Fin carrera abierto	Entrada	Bit	Verdadero si totalmente abierto
Fin carrera cerrado	Entrada	Bit	Verdadero si totalmente cerrado
Barrera infrarrojos	Entrada	Bit	Verdadero si se detecta obstáculo

Tabla 2.15 Valores propios de la puerta del garaje

2.2.2.15 Verja de entrada

Es una verja de movimiento horizontal, única en la vivienda, que permite el acceso a su perímetro exterior. En una situación normal se puede abrir únicamente a través del mando de control remoto. Solo es posible saber si el motor de la verja está activo, tanto para abrir como para cerrar, porque se señala con un indicador rojo. Los sensores (entradas) no dan señales visibles.

Etiqueta	Tipo E/S	Tipo	Descripción
Abrir	Salida	Bit	Abre la verja de entrada (domina si están activos el bit de arriba y abajo a la vez)
Cerrar	Salida	Bit	Cierra la verja de entrada
Fin carrera abierto	Entrada	Bit	Verdadero si totalmente abierto
Fin carrera cerrado	Entrada	Bit	Verdadero si totalmente cerrado
Barrera infrarrojos 1	Entrada	Bit	Verdadero si se detecta obstáculo (cara exterior de la verja)
Barrera infrarrojos 2	Entrada	Bit	Verdadero si se detecta obstáculo (cara interior de la verja)
Barrera infrarrojos 3	Entrada	Bit	Verdadero si se detecta obstáculo (en la cara interior, donde la verja quedaría situada cuando es abierta)

Tabla 2.16 Valores propios de la verja de entrada

2.2.2.16 Mando de control remoto programable de 8 botones

Es un mando de control remoto programable, con 8 botones a los que se les puede asignar una función. En situaciones normales no se usa (está preparado para uso con la consola integrada de automatización doméstica), aunque se puede preparar para uso externo, como se verá más adelante. Este no es el mando de control remoto al que se hace referencia en otras entradas de esta misma subsección.

Etiqueta	Tipo E/S	Tipo	Descripción
Botón 1	Salida	Bit	Verdadero si el botón está pulsado
Botón 2	Salida	Bit	Verdadero si el botón está pulsado
Botón 3	Salida	Bit	Verdadero si el botón está pulsado
Botón 4	Salida	Bit	Verdadero si el botón está pulsado
Botón 5	Salida	Bit	Verdadero si el botón está pulsado
Botón 6	Salida	Bit	Verdadero si el botón está pulsado
Botón 7	Salida	Bit	Verdadero si el botón está pulsado
Botón 8	Salida	Bit	Verdadero si el botón está pulsado

Tabla 2.17 Valores propios del mando de control remoto programable de 8 botones

2.2.3 Comunicaciones con el exterior de la simulación

Todo lo descrito anteriormente sobre el simulador *Home I/O* estaría muy bien como una colección de simples curiosidades si se diera el caso de que no fuera posible extraer estos valores, o no hubiera una forma relativamente sencilla de que *Home I/O* pudiera comunicarse con el exterior. No obstante (y como cabe esperar), efectivamente gran parte del valor de utilizar este simulador reside en sus capacidades de intercambio de datos con tecnologías externas. En este apartado se hace referencia a las posibilidades que *Home I/O* permite realizar estos intercambios de datos.

2.2.3.1 Modo de funcionamiento de los dispositivos

Cada dispositivo de la vivienda simulada de *Home I/O* permite su funcionamiento de uno de los tres modos que se describen a continuación [22]:

- *Wired* (modo cableado): Modo de funcionamiento que viene por defecto al lanzar una nueva simulación en la vivienda. Todos los dispositivos tienen su funcionamiento normal, el esperable cuando se adquiere una vivienda sin ningún tipo domótica ni automatización. Su etiqueta es de color rojo;
- *Wireless* (modo inalámbrico): Modo de funcionamiento, sin cables, que permite su uso para trabajo a partir de la creación de escenas sencillas a través de un dispositivo tipo Tablet de automatización, simulada, que es accesible desde el propio *Home I/O*. Al no ser de interés para este trabajo, este modo queda como mera curiosidad). Su etiqueta es de color verde;
- *External* (modo externo): El acceso al dispositivo queda abierto y disponible para lectura y escritura (solo aquellos posibles) con otras tecnologías externas, como *Connect I/O* (de la que se habla brevemente más adelante), o uso a través del SDK, que es lo que interesa. Su etiqueta es de color azul.

En la figura a continuación se muestra cómo quedaría un dispositivo conectado en cada uno de los modos anteriores:

Figura 2.8 Dispositivos de *Home I/O* en los diferentes modos de funcionamiento disponibles

2.2.3.2 Conexión con Connect I/O

Home I/O se distribuye en conjunto con *Connect I/O* [23], otro software (complementario a este) que se ofrece a través de una *IDE* gráfica y proporciona un lenguaje visual basado en bloques. Este software permite la programación directa de situaciones en *Home I/O*, así como generar en él estrategias de control o análisis de datos; o puede actuar como pasarela de integración con otras tecnologías de automatización, como autómatas programables, microcontroladores, o Modbus. Dado que se pretende hacer una integración directa con *MATLAB*, desde donde se van a leer y escribir los datos a través del *SDK* de *Home I/O*, este software no resulta de interés para el trabajo que se desarrolla.

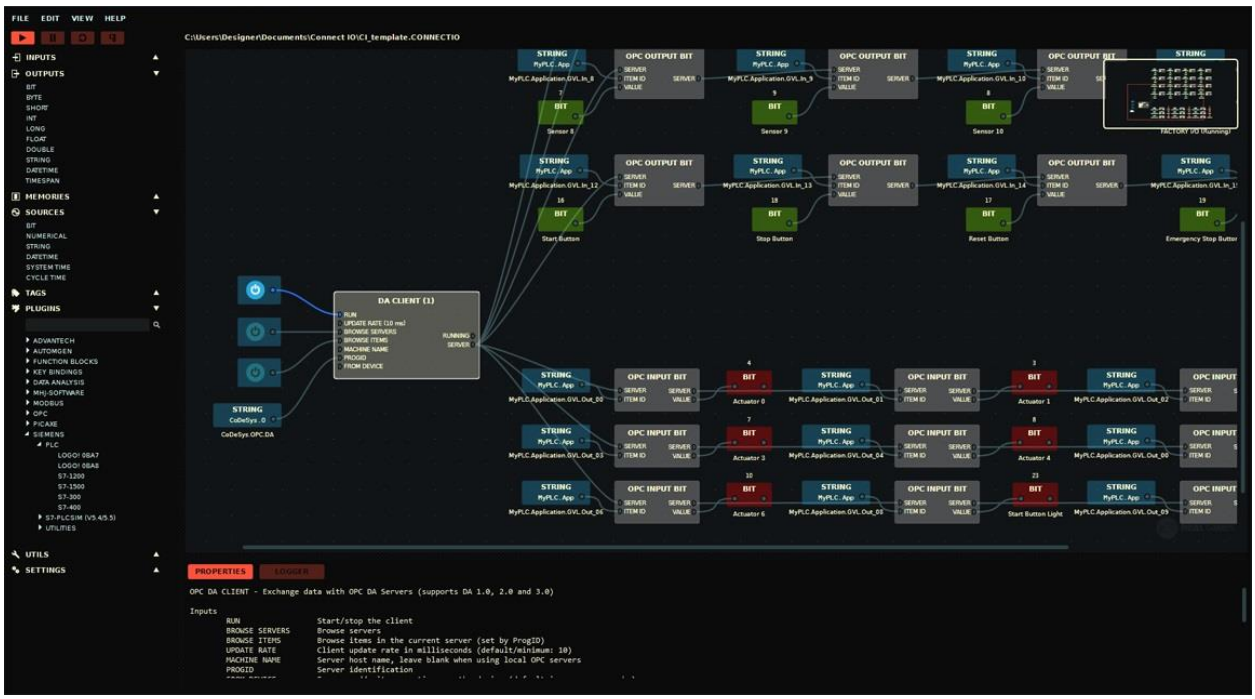


Figura 2.9 Imagen de ejemplo de la interfaz de *Connect I/O*, disponible en su documentación

2.2.3.3 Utilización del Kit de Desarrollo Software (SDK) de Home I/O

Además de la utilización a través de *Connect I/O* como pasarela de intercambio, el software *Home I/O* permite las comunicaciones con otras tecnologías a través de su SDK, siempre y cuando estas tecnologías externas admitan librerías *DLL* de *Windows* ensambladas utilizando el *Framework .NET 2.0* [24] . Esto permite acceder a los valores de la vivienda simulada desde cualquier lenguaje compatible (la página web indica *Python* y *Scratch*), siempre y cuando los dispositivos de *Home I/O* estén marcados en modo *External*.

La comunicación entre los procesos de *Home I/O* y la tecnología externa en uso se hace a través de la librería *EngineIO.dll*, que consiste en un mapa de memoria en que cada punto de acceso se divide de la siguiente forma, en una convención que han tomado como criterio de desarrollo:

- Entradas (*Inputs*): Aquellas que se registran bien por acciones del usuario (como accionar un interruptor) o por datos recogidos por los sensores instalados en la vivienda (como las temperaturas en cada estancia);
- Salidas *(Outputs*): Acciones de control que se producen como consecuencia del cambio en los valores de las entradas (como encender las luces si se acciona un interruptor), o por una decisión externa (encender un calefactor, sin otra motivación);
- Registros (*Memories)*: Valores relacionados con la simulación, de uso interno por el programa de Home I/O, que se publican siempre (no tienen modos de funcionamiento, o siempre funcionan en modo *External*) y pueden ser de utilidad para los programas externos.

Cada punto de acceso tiene su propia dirección de memoria al que se puede acceder según el tipo de dato. Los únicos casos en los que se permite el cambio de valores de la vivienda son solo aquellos en que los dispositivos aparezcan registrados como *Outputs*.

Hay 10 tipos de datos definidos en las librerías (que también aparecen en la documentación), pero de ellos solo se utilizan los siguientes:

- Bit: Valores binarios enteros (booleanos), que toman el valor de 1 (Verdadero) o 0 (Falso);
- Flotante (*Float*): Valores que pueden ser decimales, con un rango de -3.403×10^{38} a 3.403×10^{38}. Normalmente no es necesario tanto rango, porque la mayoría de entradas y salidas flotantes están limitadas al rango entre 0 y 10; y los restantes valores flotantes son temperaturas en grados Celsius o en Kelvin, que por sentido común ni siquiera llegan a 100 ºC (373.25 K);
- Fecha y hora (*DateTime*): Valores que representan un instante de tiempo, con microsegundos, con un rango rango desde medianoche del 1 de enero del año 1, hasta fin de diciembre de 9999.

Además de la librería *EngineIO.dll* se incluyen en la descarga del *SDK* unas ayudas de carácter didáctico como son un extenso fichero de ayuda (que sirve fundamentalmente para los lenguajes de programación *C#*, *Visual Basic* y *C++*, y no tanto para *MATLAB* porque usa sus propias definiciones en algunas situaciones), varios proyectos de ejemplo en C#, y una herramienta de apoyo y visualización llamada *Engine I/O Explorer*, para facilitar la familiarización con las variables de la vivienda y hacer más sencilla su visualización en tiempo real.

Por otra parte, más allá de los recursos de programación integrados en el *SDK*, la página web de Home I/O ofrece una interfaz lista para su utilización para *Scratch 2* y *Scratch 3*, y para versiones de *Python* comprendidas entre 3.4 y 3.7 (este último, a través de la librería externa *pythonnet*).

En la utilización del *SDK* de *Home I/O* se pide en la documentación prestar especial atención a lo siguiente:

- La clase *MemoryMap* (provista por el *SDK*), es de tipo *Singleton*, lo que implica que debería existir una única instancia de la misma en cualquier momento. Esta instancia es una copia en caché del archivo del mapa de memoria;
- La sincronización de esta copia en caché con el mapa de memoria se hace a través del método *MemoryMap.Instance.Update()*, por lo que este método debe llamarse cada vez que se quiera acceder a la última información publicada en los puntos de acceso o recibir las notificaciones de eventos por cambio en el valor de las variables.

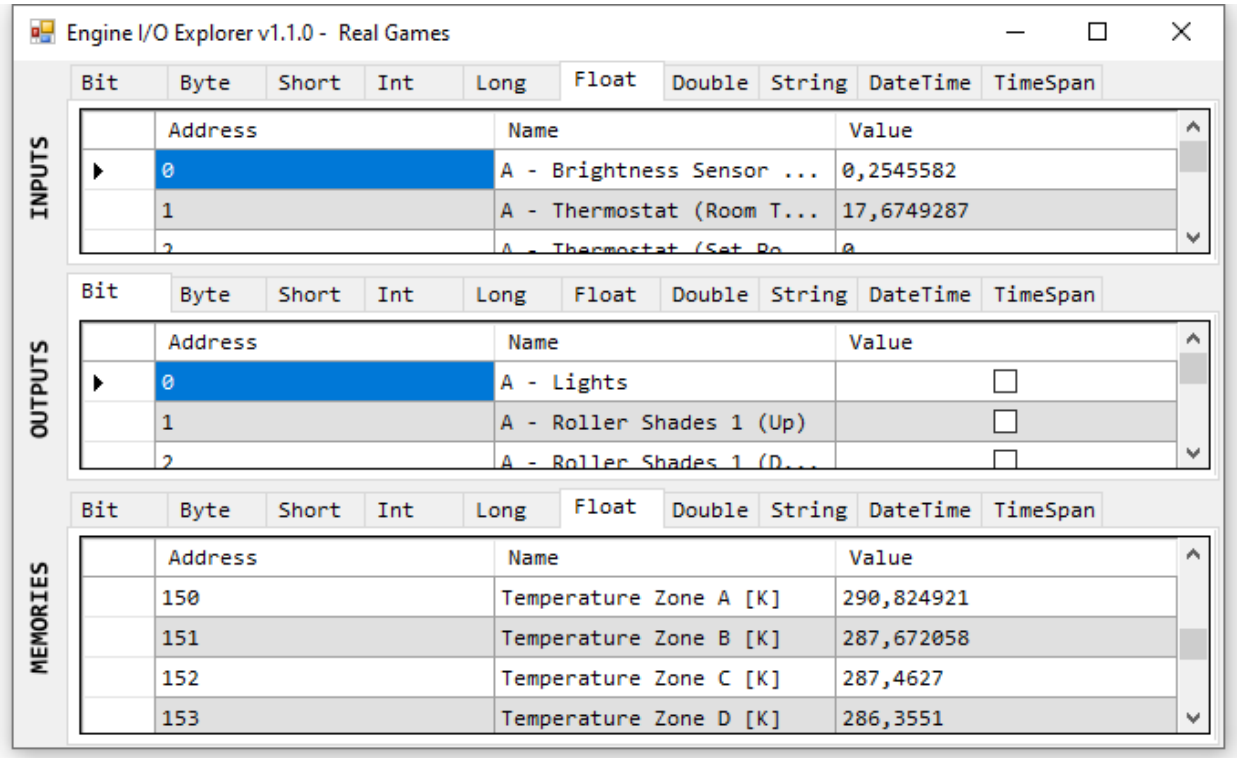


Figura 2.10 Ventana de *Engine I/O Explorer* con algunos datos en tiempo real de la vivienda de *Home I/O*

El resto de métodos y funciones provistas en el *SDK* de *Home I/O* serán utilizados cuando sean necesarios (por ejemplo, la obtención de un dato y se describirán más adelante cuando se requiera su utilización.

2.2.4 Lista completa de dispositivos en Home I/O

En la página de direcciones de memoria de *Home I/O* [25] se ofrece una lista completa de dispositivos instalados en la vivienda (y *Memories*, que no están físicamente instalados pero pueden ser de utilidad) con su nombre, tipo de variable, dirección de memoria y la estancia en que se encuentran. Las entradas binarias (de tipo *Bit*) trabajan internamente como un relé, por lo que vienen acompañadas de si son de tipo Normalmente Abierto (*Normally Open*, NO) o Normalmente Cerrado (*Normally Closed*, NC). En una tabla posterior viene una referencia de consumo energético de todos los dispositivos instalados en la casa. Nótese que todos los dispositivos que consumen potencia en este hogar simulado son de tipo *Output*, y que se da la potencia de consumo de forma simplificada. Esto quiere decir que la potencia de un calefactor encendido en tipo binario es la misma que la potencia que consume el mismo calefactor, trabajando encendido con una entrada tipo flotante a la potencia máxima que permite (10 V). El tratamiento que *Home I/O* da a estos dispositivos que trabajan tanto en modo binario como en modo flotante es el de dos entradas completamente diferentes, con su propio tipo (una es *Bit* y la otra es *Float*) y direcciones de memoria independientes.

Para adelantar parte del trabajo que corresponde a facilitar a *MATLAB* la importación de los datos, se ha trabajado en confeccionar una tabla que incluye toda la información proporcionada en esta página (tipo de contactos para las entradas binarias y consumo de potencia para las salidas que lo tengan), con los nombres originales, preparando para el trabajo en el desarrollo del programa de comunicación. Esta tabla se presentará totalmente en inglés, que es la lengua principal en el que trabaja *Home I/O* y el programa que se desarrollará para intercambio de datos y se podrá consultar en el Anexo A. La única corrección que se ha hecho a los nombres que aparecen en la mencionada página de direcciones de memoria es la corrección de erratas tipográficas en algunas entradas.

Nótese que no hay ningún valor único en la tabla (ni siquiera los nombres), y que el conjunto *Memory Type* y *Data Type* no dan un resultado único en la dirección de memoria, pero sí hay una dirección de memoria única para cada *Memory Type* y *Data Type*. De esto se puede deducir que un conjunto único en la tabla (para facilitar una búsqueda) lo pueden formar estas tres variables mencionadas.

3 CONEXIÓN ENTRE MATLAB Y HOME I/O

Entonces, en resumen, se tiene la necesidad de obtener y procesar información proveniente de la vivienda simulada en *Home I/O* desde *MATLAB*. Los datos disponibles de *Home I/O* se hacen visibles a tecnologías exteriores a través de su *SDK*, al que es posible acceder simplemente haciendo uso de la librería *.NET* conocida como *EngineIO.dll* provista por los propios desarrolladores del hogar virtual. Se han preparado todos los puntos de acceso de *Home I/O* en un formato sencillo de procesar (guardados en un documento de *Microsoft Office Excel*, *homeio_full.xlsx*). Queda por tanto hacer el resto del trabajo desde *MATLAB*, establecer un enlace entre ambos que sea funcional e intercambiar los datos de trabajo, a ser posible, en unas condiciones aceptables.

3.1. Material previo

Se tiene como material previo una simulación provista desde los materiales educativos de *Home I/O* (únicamente accesibles en francés) a través de Simulink con bloques propios de lectura y escritura de la vivienda simulada [26], en que se realiza un control integral de la luminosidad de la sala de estar (zona A) según la luminosidad deseada y la actual, con lo que se obtienen unos resultados como se muestra en la figura 3.1.

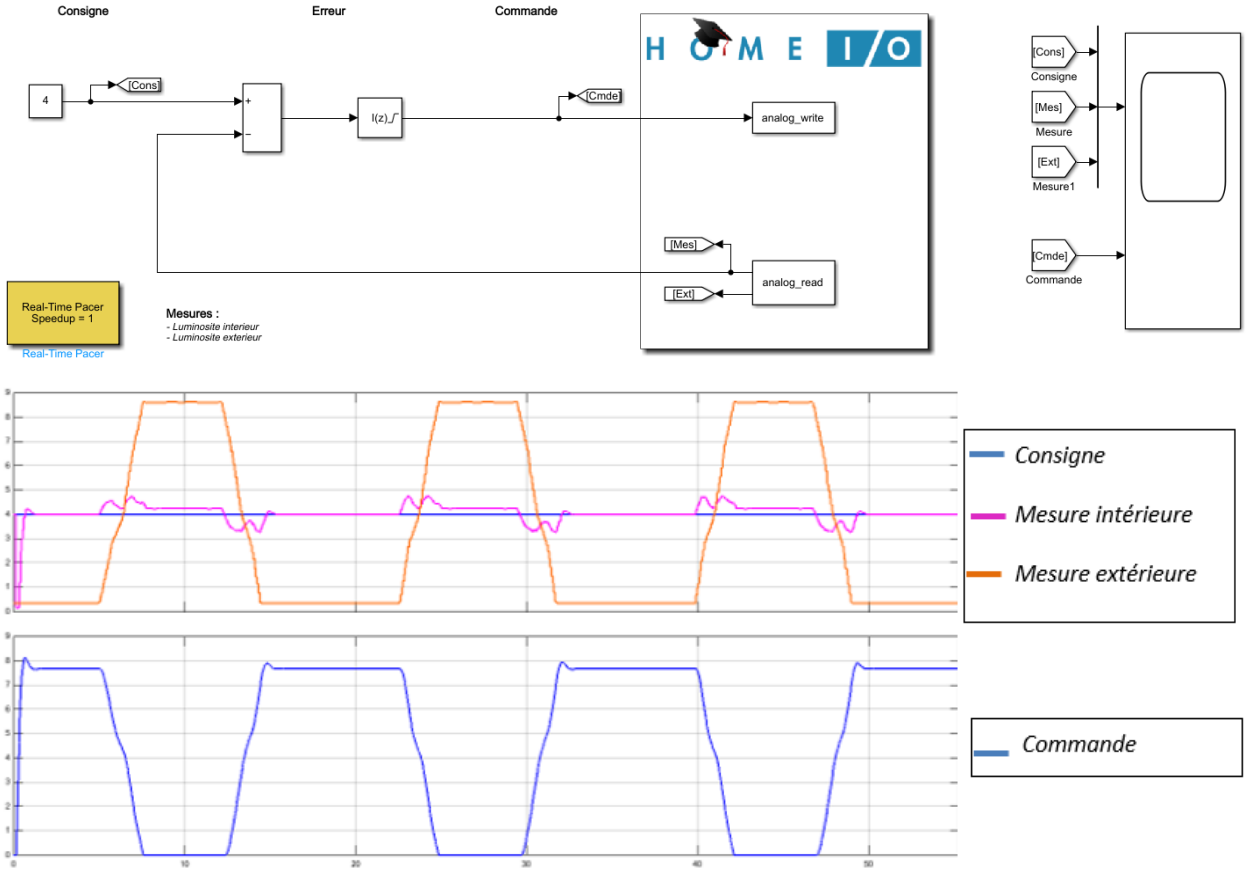


Figura 3.1 Vista del modelo de Simulink y resultados de simulación, tal como se descarga de los recursos en francés de *Home I/O*

Partiendo de esta base, se han realizado otros trabajos como el Trabajo Fin de Grado de Ezequiel Bravo del Cid [27], en que la comunicación entre las plataformas se ha realizado simplemente editando los bloques *analog_write.m* y *analog_read.m* de forma manual.

3.2 Uso del SDK de Home I/O y problemática asociada

3.2.1 Acceso y usos típicos

Para utilizar la librería *EngineIO.dll* y, de forma independiente al lenguaje que en se implemente, los casos actuales de uso más frecuente en este respecto son los que aparecen en la siguiente tabla:

Acción	Método al que llamar
Iniciar conexión con Home I/O	(automático al añadir la librería e iniciar el programa)
Actualizar datos	EngineIO.MemoryMap.Instance.Update()
Acceder a un dato *Bit*	EngineIO.MemoryMap.Instance.GetBit(<dirección>, <tipo registro>)
Acceder a un dato Float	EngineIO.MemoryMap.Instance.GetFloat(<dirección>, <tipo registro>)
Acceder a un dato DateTime	EngineIO.MemoryMap.Instance.GetDateTime(<dirección>, <tipo registro>)
Acceder a un dato (otro tipo)	EngineIO.MemoryMap.Instance.Get<tipo>(<dirección>, <tipo registro>)
Cerrar conexión con Home I/O	MemoryMap.Instance.Dispose();

Tabla 3.1 Métodos del *SDK* de la interfaz con *Home I/O* uso más frecuente

De donde se recuerda que solo se utilizan los tipos de dato *Bit, Float y DateTime* en los registros de *Home I/O*, pero hay hasta diez tipos definidos (que se ignoran al no tener uso). Por otra parte, el tipo de memoria (o de registro) no es solo poner la cadena de texto *Input*, *Output* o *Memory* tal como se ha visto en el capítulo anterior, sino que se trata de usar otras constantes definidas en la librería:

Tipo de registro	Constante que utilizar
Input	EngineIO.MemoryType.Input
Output	EngineIO.MemoryType.Output
Memory	EngineIO.MemoryType.Memory

Tabla 3.2 Constante que utilizar en el *SDK* de *Home I/O* según tipo de registro

Más aún, el acceso correcto a un dato (bajo el supuesto de que está establecida la conexión y que el dato esté adecuadamente actualizado) no devuelve directamente su valor numérico, sino que devuelve una estructura conforme a la librería, de las cuales acceder a su valor puede ser tan sencillo como acceder a su atributo Value, como se puede apreciar en la siguiente figura:

```
>> EngineIO.MemoryMap.Instance.GetFloat(1, EngineIO.MemoryType.Output)

ans =

  MemoryFloat with properties:

        Value: 7
   MemoryType: Output
      Address: 1
         Name: [1×1 System.String]
      HasName: 1

>> EngineIO.MemoryMap.Instance.GetFloat(1, EngineIO.MemoryType.Output).Value

ans =

  single

     7
```

Figura 3.2 Acceso a un dispositivo de *Home I/O* desde *MATLAB*

Pero los tipos de datos *DateTime* son diferentes. En caso de querer acceder a su valor, el programador se puede encontrar con una sorpresa:

```
>> EngineIO.MemoryMap.Instance.GetDateTime(65, EngineIO.MemoryType.Memory).Value

ans =

  DateTime with properties:

          Date: [1×1 System.DateTime]
           Day: 27
     DayOfWeek: Wednesday
     DayOfYear: 300
          Hour: 3
          Kind: Unspecified
   Millisecond: 780
        Minute: 44
         Month: 10
           Now: [1×1 System.DateTime]
        UtcNow: [1×1 System.DateTime]
        Second: 50
         Ticks: 637709030907800000
     TimeOfDay: [1×1 System.TimeSpan]
         Today: [1×1 System.DateTime]
          Year: 2021
      MinValue: [1×1 System.DateTime]
      MaxValue: [1×1 System.DateTime]
```

Figura 3.3 Intento de acceso al valor de un *DateTime*

3.2.2 Accesos a valores inexistentes sin aparición de errores

Tal como se lleva haciendo hasta ahora, para extraer un valor de *Home I/O* es necesario saber exactamente todos los valores numéricos relativos al dispositivo de la simulación se está queriendo acceder y utilizar el método adecuado de la librería *EngineIO*.dll, además de que no se proporciona gestión de errores en caso de equivocarse con el direccionamiento. Por tanto, cometer un error tipográfico (equivocarse en el tipo de dato o de memoria pero escribiendo valores válidos, o metiendo una dirección de memoria en el rango de trabajo) se devuelve un valor completamente válido sin señalizar el problema. Esta problemática se ilustra a continuación, en que no existe tal entrada de memoria registrada como un dispositivo válido en Home I/O (existe una entrada binaria con la dirección de memoria 153, que es el sensor de movimiento de la sala I, pero no la dirección de memoria 154 para entradas binarias):

```
>> EngineIO.MemoryMap.Instance.GetBit(154, EngineIO.MemoryType.Input).Value

ans =

  logical

   0
```

Figura 3.4 Acceso a un valor no registrado, sin ningún tipo de error

Acceder a valores fuera de rango, por ejemplo una dirección de memoria negativa o muy alta, sí devuelve un error al usuario.

3.2.3 Se permite la utilización de valores arbitrarios fuera de rango

Actualmente, nada limita la utilización de valores arbitrarios para aquellos valores clasificados como *Output* y *Float* que dicen estar limitados en el rango de 0-10 V. De forma breve, aquí se utiliza el programa de debug *Engine I/O Explorer* para dar un valor absurdamente grande. Obsérvese que Home I/O limita internamente el valor al rango adecuado y que la potencia consumida es de 2000 W, la potencia total del calefactor. Lo mismo ocurre al darle un valor negativo, que se recorta al rango [0,10] (en este caso, a 0), pero no se muestra al usuario. Por otra parte, sí que se produce un error si se intenta poner valores que no corresponden para el tipo (por ejemplo caracteres o números fuera de los límites de un valor flotante).

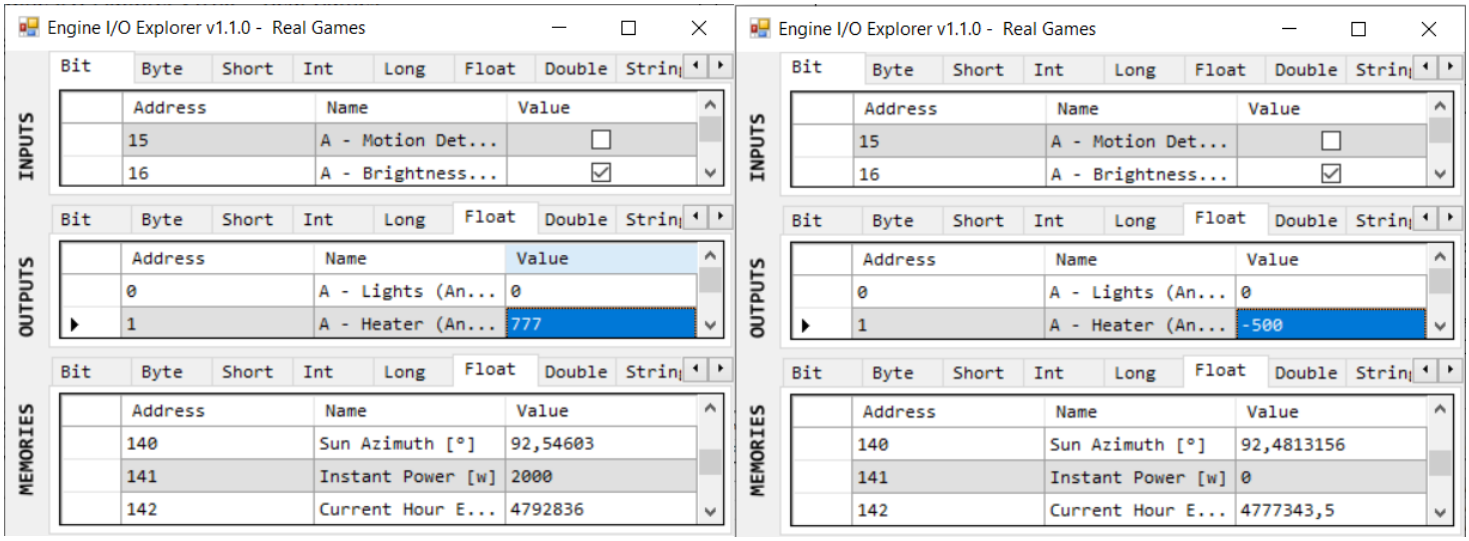


Figura 3.5 Asignación de valores absurdos a dispositivos en *Home I/O*

3.2.4 Conflictos en el acceso a valor de algunos dispositivos

Tal como se indicaba en el anterior apartado 2.2.2 (tipos de dispositivos), aquellos dispositivos que se comportan como *Output* y son *Bit* y *Float* a la vez (como las luces y los calefactores) tienen este problema:

> No se indica en la documentación, pero los valores flotantes dados a estos dispositivos toman prioridad frente a los valores de bit. Es decir, el valor binario de los dispositivos se ignora si hay un valor entre 0 y 10 para controlarlos, llegando a producirse situaciones como que, en caso de apagar calefactor (por la vía analógica) llegue a encenderse por completo al estar su bit activo. Se harán consideraciones a este respecto más adelante.

Una forma sencilla de comprobarlo es utilizando de nuevo el *Engine I/O Explorer* y encender el valor binario de un calefactor mientras el valor flotante de este aparato físico queda en un valor intermedio (7 en este caso). Se puede comprobar que, aunque esté encendido en su valor binario, consume la parte proporcional de potencia indicada en su valor flotante (1400 W). Esto ocurre igual con las luces de las estancias.

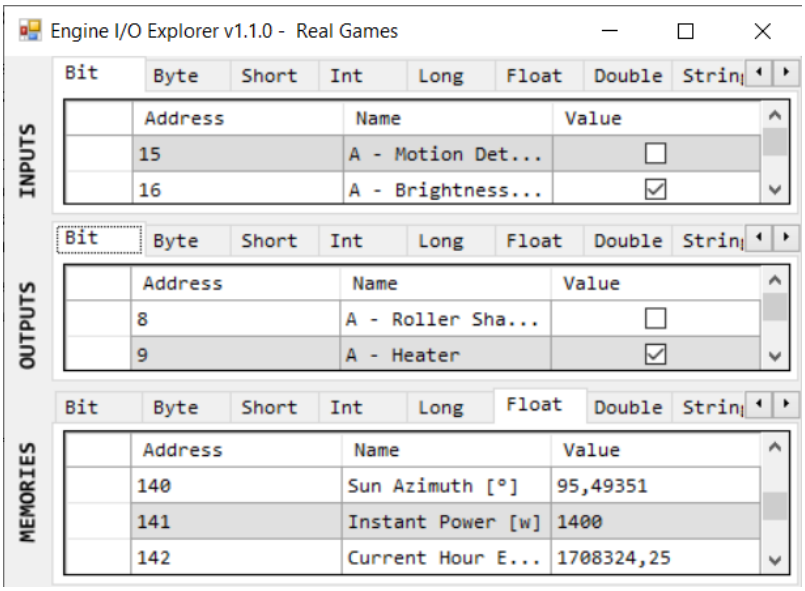


Figura 3.6 Calefactor encendido en binario, limitado por su valor flotante

No es fácil demostrar con imágenes la segunda afirmación (que al apagar el valor flotante mientras el valor booleano se mantiene en verdadero, el dispositivo pasa a estar encendido), pero sí es fácil entender el razonamiento lógico del porqué.

3.2.4 Se permite escribir valores en *Inputs* y *Memories*, pero son ignorados por el programa

Mientras que los únicos dispositivos que experimentan un cambio al alterar su valor son los dispositivos tipo *Outputs* (lo cual tiene lógica porque son acciones de control y son las acciones que se toman en respuesta de un sensor o un interruptor siendo accionado), tanto el *Engine I/O Explorer* como desde la *API* se permite cambiar el valor de cualquier entrada de memoria catalogada como *Input* o *Memory*. No obstante, el cambio será ignorado por el simulador *Home I/O*, que sobrescribirá este valor en el siguiente paso de simulación sin haber hecho uso de este valor sobrescrito. La única excepción a esto son los interruptores binarios de luz (no los atenuadores), que mantienen registrados el cambio pero igualmente el programa lo ignora.

Esto tal vez pudiera entenderse físicamente como, para los valores *Input*, si se quisiera forzar una lectura falsa en algún sensor, quizá como efecto de partícula ionizante de alta energía proveniente del exterior causado por un rayo cósmico [28], aunque estos eventos son relativamente extremadamente poco frecuentes en la Tierra (al menos en lugares dentro de la electrónica en que tengan relevancia). No obstante, sí tiene menos sentido con los valores *Memory*, que son modificables por el usuario dentro de *Home I/O* (con unos límites), pero son ignorados igualmente por el programa, por lo que cualquier cambio en ellos debe hacerse manualmente en él.

Este problema puede verse de forma simplificada en la siguiente figura de código, en la que se intenta acceder y cambiar el valor de la temperatura de la sala D (cocina) y que al recargar los valores de *Home I/O* de vuelta se pierde el cambio que se ha intentado implementar. Para simplificar su entendimiento, los pasos que se siguen aquí son los siguientes:

1. Refresco de los valores de *Home I/O* para tener los valores más recientes y acceso al dato referido (*MemoryFloat* de dirección de memoria 153);
2. Cambio del valor y actualización para que se refleje en *Home I/O*. Acceso una segunda vez para comprobar que el valor aparece registrado;
3. Refresco tras la espera de al menos un *frame* de simulación en *Home I/O* (en que se recalculan todos los valores) y acceso de nuevo al valor para comprobar que el valor dado se pierde.

El siguiente código de prueba permite ejemplificar este problema:

```
>> EngineIO.MemoryMap.Instance.Update();
>> test = EngineIO.MemoryMap.Instance.GetFloat(153, EngineIO.MemoryType.Memory)

test =

  MemoryFloat with properties:

        Value: 283.2928
   MemoryType: Memory
      Address: 153
         Name: [1×1 System.String]
      HasName: 1

>> test.Value = 100; EngineIO.MemoryMap.Instance.Update();
>> test = EngineIO.MemoryMap.Instance.GetFloat(153, EngineIO.MemoryType.Memory)

test =

  MemoryFloat with properties:

        Value: 100
   MemoryType: Memory
      Address: 153
         Name: [1×1 System.String]
      HasName: 1

>> EngineIO.MemoryMap.Instance.Update();
>> test = EngineIO.MemoryMap.Instance.GetFloat(153, EngineIO.MemoryType.Memory)

test =

  MemoryFloat with properties:

        Value: 283.2920
   MemoryType: Memory
      Address: 153
         Name: [1×1 System.String]
      HasName: 1
```

Figura 3.7 *Home I/O* ignora los cambios a tipos de registros que no sean *Outputs*

O bien, podría reproducirse mediante el siguiente código

```
% test_error comprobación del error de cambio de Inputs o Memories
% Acceso 1: obtención del valor inicial
EngineIO.MemoryMap.Instance.Update();
test.Name
test = EngineIO.MemoryMap.Instance.GetFloat(153, EngineIO.MemoryType.Memory)


% Acceso 2: cambio del valor
test.Value = 100; EngineIO.MemoryMap.Instance.Update();
test = EngineIO.MemoryMap.Instance.GetFloat(153, EngineIO.MemoryType.Memory)

% Acceso 3: espera de 2 frames (2/60 s) (el actual en que permanece el
% cambio y el siguiente en que recalcula) y vuelta a lo anterior a refrescar
pause(2/60)
EngineIO.MemoryMap.Instance.Update();
test = EngineIO.MemoryMap.Instance.GetFloat(153, EngineIO.MemoryType.Memory)
```

A la vista de la problemática encontrada, que puede darse al trabajar con el simulador de vivienda *Home I/O*, se propone el desarrollo de una interfaz en *MATLAB* (que se materializará como una clase) que resuelva los citados problemas y facilite su uso de cara al usuario.

3.3 Arquitectura software (simplificada)

La comunicación que se establece entre *MATLAB* y *Home I/O* (o entre Home I/O y cualquier otro programa que utilice su SDK), ya sin necesidad de ninguna explicación adicional, sigue un esquema similar al que se muestra a continuación:

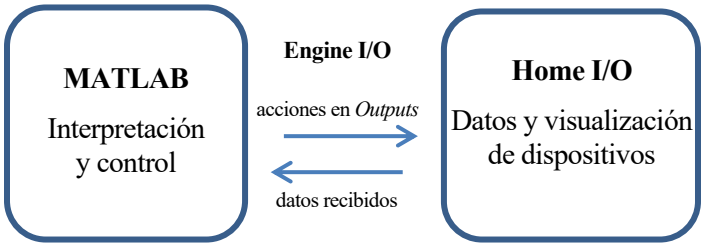


Figura 3.8 Esquema simplificado de comunicación entre *MATLAB* y *Home I/O*

A través de la librería *Engine I/O* se posibilita la comunicación bilateral entre ambas piezas de software. Es ya función de MATLAB (en este caso) trabajar con los datos que reciba y procesarlos según lo que se necesite.

3.4 Un ejemplo sencillo de uso del SDK de Home I/O en MATLAB (sin Simulink)

En la situación actual, un ejemplo sencillo de uso de las comunicaciones entre *MATLAB* y *Home I/O* puede ser similar al que se ofrece como ejemplo para otros lenguajes: encender y apagar la luz del salón (zona A). Este ejemplo va un poco más allá y comprueba la potencia consumida y el cambio de luminosidad tras cambiar el valor del calefactor del salón a encendido (en modo binario) y encender las luces (en modo flotante) a un valor de 7. A este código se le ha añadido una búsqueda del proceso "*Home IO.exe*" [29] para garantizar que el simulador de vivienda se encuentra en ejecución.

(El código puede aparecer algo deformado debido al efecto de copiarlo y pegarlo en el procesador de textos, pero es correcto y puede ser igualmente copiado y pegado a un script de *MATLAB* funcional).

```
%% Add Engine I/O library and previous checks
path = strcat(pwd,'\EngineIO.dll');
NET.addAssembly(path);

% Home IO process MUST be running at this point or everything will return zero values
% https://stackoverflow.com/questions/858301/detect-matlab-processes-from-within-
matlab
% Wolfie/Edric's answers edited to check Home IO process
% Runs on Windows only
[~,w] = dos( 'tasklist /fi "IMAGENAME eq Home IO.exe" /fo "csv"' );
num_homeios = length( regexp( w, '"Home IO.exe"' ) );
if num_homeios == 0
    error("Home I/O process not found. Unable to proceed.");
elseif num_homeios ~=1
    % Home I/O normally blocks more than one instance, but to be extra safe
    error("You must be running exactly 1 instance of Home I/O. You are running
%d.",num_homeios)
end
clear("num_homeios");

fprintf("\nHome I/O process found. Remember to set all necessary inputs and outputs
to External mode.\n\n")

% Mandatory update prior to reading or fist read will go wrong.
EngineIO.MemoryMap.Instance.Update();

%% Program starts here
% Read operations
livingLightSensor = EngineIO.MemoryMap.Instance.GetFloat(0,
EngineIO.MemoryType.Input).Value
powerConsumed = EngineIO.MemoryMap.Instance.GetFloat(141,
EngineIO.MemoryType.Memory).Value

% Write operations must retrieve the object first, then set the value!
% Otherwise an error will be thrown!
livingLights = EngineIO.MemoryMap.Instance.GetFloat(0, EngineIO.MemoryType.Output);
livingLights.Value = 7;

livingheat = EngineIO.MemoryMap.Instance.GetBit(9, EngineIO.MemoryType.Output);
livingheat.Value = 1;

% Update
EngineIO.MemoryMap.Instance.Update();

% Wait some frames and update again to make sure it takes effect
% No update = experiment breaks
```

```
pause(10/60)
EngineIO.MemoryMap.Instance.Update();

% Read operations again
livingLightSensor = EngineIO.MemoryMap.Instance.GetFloat(0,
EngineIO.MemoryType.Input).Value
powerConsumed = EngineIO.MemoryMap.Instance.GetFloat(141,
EngineIO.MemoryType.Memory).Value
```

De donde se puede comprobar que tiene el efecto deseado en la simulación de *Home I/O*. Obsérvese que el calefactor y las luces están encendidas, y también el cambio en el consumo instantáneo de la vivienda de un valor nulo a 2042 W), lo cual también se puede observar a través de los valores de salida en *MATLAB*:

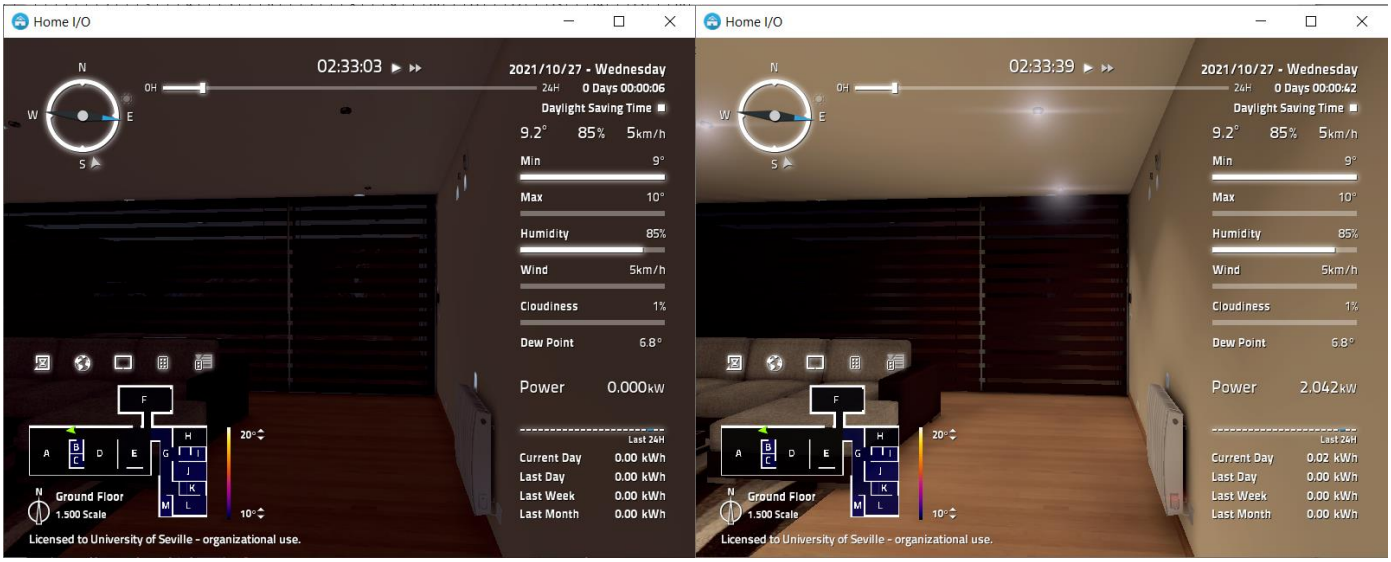


Figura 3.9 Control sencillo y directo de dispositivos de *Home I/O* desde *MATLAB*

```
>> communications_script
Home I/O process found. Remember to set all necessary inputs and outputs to External mode.

livingLightSensor =
  single
    0.0850

powerConsumed =
  single
     0

livingLightSensor =
  single
    3.5850

powerConsumed =
  single
        2042
```

Figura 3.10 Efecto del cambio de valores del ejemplo, vistos desde *MATLAB*

4 INTERFAZ DESARROLLADA PARA COMUNICACIONES MATLAB - HOME I/O

La solución que se propone a los problemas detectados, así como una simplificación para el usuario cuando se trate de utilizar *Home I/O*, es una clase desarrollada en *MATLAB* [30] que provee, de forma eficiente, de una interfaz de comunicaciones entre ambos programas, sencilla y eficiente, la cual se pasa a describir a continuación. Para facilitar la consulta, una lista de documentación utilizada de MATLAB se dispondrá de forma adecuadamente tematizada al final de este capítulo y en el texto se mencionarán las entradas estrictamente necesarias.

4.1 Descripción general

La clase de comunicación funciona de la siguiente forma:

1. Inicialización. Lectura de variables aportadas;
2. Conexión a *Home I/O* (a través del *DLL*, no comprueba todavía su estado online);
3. Lectura de la tabla en Excel que tiene todos los dispositivos en el atributo *BaseData*. Categorizaciones adicionales para mejora en la velocidad posterior de procesamiento;
4. Recálculo de la potencia en dispositivos que trabajan en el rango 0-10 V (división entre 10) de modo que la potencia utilizada sea la potencia por el valor de la salida en que se encuentre;
5. Separación de los datos completos en categorías para que sean de fácil acceso, todas guardadas en atributos del objeto;
6. Creación de un nuevo atributo, inicialmente copia de *BaseData* que sea responsable de almacenar además los datos generales junto a los valores leídos (en *FullData*);
7. Si Home I/O está abierto, actualizar todos los valores;
8. Si se especifica en las configuraciones, abrir escuchas hacia los eventos de Home I/O para actualizar los valores cambiados en cuanto se produzca el cambio;
9. Si se especifica la instalación de un temporizador para actualizaciones regulares, prepararlo y lanzarlo.

En caso de que se haya puesto un temporizador, los valores se irán actualizando de forma automática de fondo a través de la actualización de valores de *Home I/O*, que llamará a los respectivos eventos y lo procesará (o, en caso de que los eventos no se encuentren activos, irá llamando a la función de volcado de datos completa).

Una vez cargado el objeto, se tienen a disposición numerosos atributos y métodos para hacer el trabajo con *Home I/O* más sencillo al usuario.

Como se va a trabajar alterando directamente valores en esta clase y se van a utilizar manejadores de eventos, esto obliga a que la mencionada clase de comunicaciones sea de tipo *handle* [31] [32] [33] [34] en MATLAB o no funcionará.

4.2 Lista de atributos

Como se comentaba antes, la mayoría de los atributos de esta clase de comunicación son subconjuntos de la lista completa de dispositivos, los cuales se han elegido de modo que facilite encontrar los dispositivos que se requieran en cada momento según las necesidades que tenga el usuario, o sean utilizados por algunos métodos usados en esta misma clase. Por tanto, puede que se produzca una multiplicidad de datos guardados que vienen a representar las mismas realidades, pero de esta manera se cumplen dos requisitos de interés:

1. Se hace más sencillo de seguir y acceder por parte del usuario, puesto que siguen ordenaciones lógicas sencillas de memorizar;
2. Se procesan unas cantidades menores de datos, que ayudan en la eficiencia y la velocidad de ejecución, permitiendo tasas de refresco más altas en caso de ser necesarias.

4.2.1 Atributos que contienen dispositivos.

Estos atributos son:

- **BaseData**: Contiene la lista completa de dispositivos, junto a las columnas *VarType* y *RowID*, que permiten trabajar de forma más rápida en otros métodos de la clase;
- **FullData**: Es una combinación de *BaseData* con los valores obtenidos. Único atributo con los valores de los dispositivos, al que se accede por filas, para que sea rápido de utilizar y sencillo de entender. También se puede ver directamente para comprobar en directo los valores y sus cambios, en caso de que se tenga el temporizador activado para actualizar los valores;
- **Types**: Contiene los dispositivos filtrados y clasificados según *Data Type* y *Memory Type*. Sus componentes son auto explicativos. El subconjunto *OutputDateTimes* queda vacío (no hay dispositivos de esa clase) de forma intencionada.
    i. **Types.Inputs**
    ii. **Types.Outputs**
    iii. **Types.Memories**
    iv. **Types.Bools**
    v. **Types.Floats**
    vi. **Types.DateTimes**
    vii. **Types.InputBools**
    viii. **Types.InputFloats**
    ix. **Types.InputDateTimes**
    x. **OutputBools**
    xi. **OutputFloats**
    xii. **OutputDateTimes**
    xiii. **MemoryBools**
    xiv. **MemoryFloats**
    xv. **MemoryDateTimes**
- **Zones**: Contiene los dispositivos filtrados por estancias.
    i. **Zones**.**ZoneA** … hasta **Zones.ZoneO**
    ii. **Zones.ZoneNone** para dispositivos que no tienen asignada una estancia (como puede ser el caso de muchas *Memories*, por ejemplo y por si revistiera utilidad)

- **Devices**: Todos los dispositivos filtrados por tipo de dispositivo, separados según tipo de funcionamiento.

  (Dispositivos tipo *Inputs*)

  i. **Devices. LightSwitches**
  ii. **Devices. UpDownSwitches**
  iii. **Devices.LightDimmersUpDown**
  iv. **Devices.DoorDetectors**
  v. **Devices.MotionDetectors**
  vi. **Devices.BrightnessSensorsBool**
  vii. **Devices.SmokeDetectors**
  viii. **Devices.AlarmKeyPadArmed**
  ix. **Devices.GarageDoorSensors**
  x. **Devices.EntranceGateSensors**
  xi. **Devices.RemoteButtons**
  xii. **Devices.BrightnessSensorsFloat**
  xiii. **Devices.ThermostatTemperatures**
  xiv. **Devices.ThermostatSetPoints**
  xv. **Devices.RollerShades**
  xvi. **Devices.DateAndTimeInput**

  (Dispositivos tipo *Outputs*)

  xvii. **Devices.LightsBool**
  xviii. **Devices.LightsFloat**
  xix. **Devices.RollerShadesUpDown**
  xx. **Devices.HeatersBool**
  xxi. **Devices.HeatersFloat**
  xxii. **Devices.Sirens**
  xxiii. **Devices.AlarmKeyPad**
  xxiv. **Devices.GarageDoorOpenClose**
  xxv. **Devices.EntranceGate**

  (Dispositivos tipo *Memories*)

  xxvi. **Devices.DLST**
  xxvii. **Devices.TimeScale**
  xxviii. **Devices.LatLongitude**
  xxix. **Devices.EnvironmentValues**
  xxx. **Devices.InstantPower**
  xxxi. **Devices.CurrentPower**
  xxxii. **Devices.LastPower**
  xxxiii. **ZoneTemperatures**
  xxxiv. **Devices.DateAndTimeMemory**

Se ha considerado dejar en la misma categoría las acciones de activación y desactivación tanto de inputs como de outputs para hacer más sencillo el trabajo al programa. También, la categoría *EnvironmentValues* es la única variopinta, que recoge todos los datos ambientales y de simulación en lo que respecta al entorno de la vivienda (direcciones *Memory* flotantes desde la 132 a la 140).

- **Special**: Contiene subconjuntos que pueden ser útiles desde el punto de vista de la programación debido a que pueden requerir algún tipo de tratamiento especial o particular.
    i. **Special.InputsNC**
    ii. **Special.InputsNO**
    iii. **Special.Inputs10V**
    iv. **Special.Outputs10V**
    v. **Special.ConflictInputs** (entradas que no pueden ser asignadas a la vez, como un interruptor arriba y abajo)
    vi. **Special.ConflictOutputs** (salidas que no pueden ser asignadas a la vez, como un calefactor en modo binario y en modo flotante)

4.2.2 Atributos de configuración y comunicaciones

- **Comm**; contiene los valores de comunicación e interrupciones consigo mismo: los manejadores de eventos y el temporizador, que se asignarán si así queda configurado.
    i. **Comm.lh1** (manejador para eventos de cambio en dispositivos *Inputs*)
    ii. **Comm.lh1** (manejador para eventos de cambio en dispositivos *Outputs*)
    iii. **Comm.lh1** (manejador para eventos de cambio en dispositivos *Memories*)
    iv. **Comm.t** (objeto timer)
- **Config**: las configuraciones entregadas al objeto, por si es necesario guardarlo y restaurarlo posteriormente.
    i. **Config.engineio_path** (ruta de acceso a la librería *EngineIO.dll*)
    ii. **Config.data_path** (ruta de acceso al documento Excel con la tabla de dispositivos)
    iii. **Config.set_timer** (si se desea ejecutar un temporizador de fondo)
    iv. **Config.set_listener** (si se desea usar los manejadores de eventos, que es más rápido)

Todos los atributos se encuentran inicialmente vacíos, por lo que si están solos simplemente se declaran (como es en los casos *BaseData* y *FullData*); o si forman parte de una estructura, se declara una estructura vacía. Añadir miembros de estructuras en el apartado de atributos de una clase en MATLAB causa errores porque en este lugar se requeriría que estuvieran completamente formadas [35].

4.3 Lista de métodos

A continuación, se relaciona una lista completa de los métodos incluidos en la clase desarrollada para este trabajo. Para facilitar la lectura, se seguirá una estructura común para todos métodos, consistente en un título de tercer orden (nueva subsección) con el nombre del método, seguido de una lista como la que procede:

- **Nombre del método:**
- **Visibilidad** (público o privado):
- **Entradas requeridas:**
- **Entradas opcionales:** (los argumentos por nombre y valor exigen primero una cadena de caracteres que indique la variable que se quiere asignar)
- **Funcionamiento esperado:**
- **Salidas esperadas:**
- **Código:** (copiado desde la fuente, puede haber problemas de formato en este documento, pero funciona bien en *MATLAB*)

Para facilitar la consulta, una lista de documentación utilizada de *MATLAB* se dispondrá de forma adecuadamente tematizada al final de este capítulo.

4.3.1 Constructor

- **Nombre del método:** HomeIO
- **Visibilidad** (público o privado): público
- **Entradas requeridas**: (ninguna)
- **Entradas opcionales**: argumentos dados por nombre y valor:
    - *engineio_path*: ruta a la librería del *SDK* de *Home I/O*. Por defecto, se busca el archivo *EngineIO.dll* dentro del mismo directorio.
    - *data_path*: ruta al fichero *Excel* con la lista completa de dispositivos. Por defecto, se busca el archivo *homeio_full.xlsx* dentro del mismo directorio.
    - *set_timer*: valor booleano. *Verdadero* si se quiere ejecutar un temporizador de fondo que actualice los valores del atributo *FullData*. Por defecto es *Falso*.
    - *set_listeners*: valor booleano. *Verdadero* si se quiere que cada vez que se actualicen los datos de *Home I/O* se lance un evento que actualice los valores de *FullData* que hayan cambiado desde el último refresco. Por defecto es *Verdadero*.
- **Funcionamiento esperado:** (el descrito en la sección 4.1 Descripción general)
- **Salidas esperadas:** objeto Home I/O disponible para utilizar, con los atributos descritos anteriormente y los métodos que en esta sección se están comentando.
- **Código:**

```
function obj = HomeIO(varargin)
    %HOMEIO Constructor
    %   Prepares the class by running the .NET dependency and
    %   updating the MemoryMap once if Home I/O is running
    %   Also allows specifying a non-default path for finding the
    %   Engine I/O DLL library and capability to update data via a
    %   timer or externally
```

```
            %% Parse inputs
            default_engineio_path = strcat(pwd,'\EngineIO.dll');
            default_data_path     = strcat(pwd,'\homeio_full.xlsx');
            default_timer         = false;
            default_listen        = true;

            p = inputParser;
            p.addParameter('engineio_path',default_engineio_path,@mustBeFile);
            p.addParameter('data_path',default_data_path,@mustBeFile);
            p.addParameter('set_timer',default_timer,@mustBeNumericOrLogical);

p.addParameter('set_listener',default_listen,@mustBeNumericOrLogical);
            p.parse(varargin{:});

            obj.Config.engineio_path = p.Results.engineio_path;
            obj.Config.data_path     = p.Results.data_path;
            obj.Config.set_timer     = p.Results.set_timer;
            obj.Config.set_listener  = p.Results.set_listener;

            %% Prepare required libraries and data
            % Link Engine I/O library now (ensures the correct path)
            obj.connectHomeIO(obj.Config.engineio_path);

            % Read and prepare all data from the table (same with path)
            obj.BaseData = obj.readExcelData(obj.Config.data_path);

            % Adapt power consumption to work with the estimatePower method
            % for outputs that are in range 0-10V
            obj.changePower10V();

            % Split data into fixed groups
            obj.splitData();

            % Add a column to store values, DateTime values must be stored
            % in numerical form (if ever rendered usable). In another
            % variable to preserve table data structure
            obj.FullData = obj.BaseData;
            obj.FullData.Value = nan(height(obj.FullData),1);

            % Check Home I/O, update, preload values
            % Just warn, in case we are using an offline object
            if obj.checkHomeIO("action","warning")
                obj.updateHomeIO();
                obj.FullData.Value = obj.readValues(obj.BaseData);

            end

            %% Event listeners and timer setup

            % Listeners if specified
            if obj.Config.set_listener
                obj.Comm.lh1 =
listener(EngineIO.MemoryMap.Instance,'InputsValueChanged',@obj.OnValuesChanged);
                obj.Comm.lh2 =
listener(EngineIO.MemoryMap.Instance,'OutputsValueChanged',@obj.OnValuesChanged);
                obj.Comm.lh3 =
listener(EngineIO.MemoryMap.Instance,'MemoriesValueChanged',@obj.OnValuesChanged)
;
            end
```

```
            % Timer if specified
            if obj.Config.set_timer
                obj.Comm.t = timer;
                obj.Comm.t.BusyMode = 'drop'; % maybe 'queue' could fit other
needs
                obj.Comm.t.ExecutionMode = 'fixedRate';
                %obj.Comm.t.TasksToExecute = 10; % test
                %obj.Comm.t.Period = 1/60;
                obj.Comm.t.Period = 1;

                if obj.Config.set_listener
                    obj.Comm.t.TimerFcn = @obj.OnTimerCall;
                    %obj.Comm.t.StopFcn  = @obj.OnTimerEnd;
                    %obj.Comm.t.ErrorFcn = @obj.OnTimerEnd;
                else
                    obj.Comm.t.TimerFcn = @obj.OnTimerCallNoListener;
                    %obj.Comm.t.StopFcn  = @obj.OnTimerEnd;
                    %obj.Comm.t.ErrorFcn = @obj.OnTimerEnd;
                end

                tic; % for performance measurements
                start(obj.Comm.t);
            end
        end
```

4.3.2 connectHomeIO

- **Nombre del método**: connectHomeIO
- **Visibilidad** (público o privado): público
- **Entradas requeridas**: (ninguna)
- **Entradas opcionales**: argumento dado por nombre y valor:
    - *engineio_path*: ruta a la librería del *SDK* de *Home I/O*. Por defecto, se busca el archivo *EngineIO.dll* dentro del mismo directorio.
- **Funcionamiento esperado**: añade al hilo de trabajo de *MATLAB* archivo de la librería *.NET* que contiene el *SDK* de *Home I/O* para funcionamiento con el simulador de vivienda. La librería queda vinculada hasta que se cierra *MATLAB*. Método estático que no necesita hacer uso de ningún atributo ni método del propio objeto.
- **Salidas esperadas**: (ninguna, la vinculación queda automáticamente realizada)
- **Código:**

```
        function connectHomeIO(varargin)
            %Connects this thread to the Engine I/O DLL library

            default_path = strcat(pwd,'\EngineIO.dll');
            p = inputParser;
            p.addOptional('engineio_path',default_path,@mustBeFile);
            p.parse(varargin{:});

            engineio_path = p.Results.engineio_path;

            % Link Engine I/O library
            NET.addAssembly(engineio_path);
        end
```

4.3.3 readExcelData

- **Nombre del método**: readExcelData
- **Visibilidad** (público o privado): público
- **Entradas requeridas**: (ninguna)
- **Entradas opcionales**: argumento dado por nombre y valor:
    - *data_path*: ruta al fichero *Excel* con la lista completa de dispositivos. Por defecto, se busca el archivo *homeio_full.xlsx* dentro del mismo directorio.
- **Funcionamiento esperado**: lee el fichero *Excel* dado en la ruta especificada como una tabla y lo carga en la variable *BaseData*. Se le añaden y calculan columnas (variables) adicionales para acelerar el trabajo de otros métodos de esta clase y simplificar la búsqueda de dispositivos por parte del usuario. Método estático que no necesita hacer uso de ningún atributo ni método del propio objeto.
- **Salidas esperadas**: devuelve el atributo *BaseData* adecuadamente cargado
- **Código:**

```
function out = readExcelData(varargin)
    %Read and preprocess all data from the Excel table to get the
    %   array to assign, typically, to obj.BaseData

    %% Parse inputs
    default_data_path     = strcat(pwd,'\homeio_full.xlsx');

    p = inputParser;
    p.addOptional('data_path',default_data_path,@mustBeFile);
    p.parse(varargin{:});

    data_path = p.Results.data_path;

    %% Read and prepare all data from the table (same with path)
    warning('OFF', 'MATLAB:table:ModifiedAndSavedVarnames')
    out = readtable(data_path,'VariableNamingRule',"modify");
    warning('ON', 'MATLAB:table:ModifiedAndSavedVarnames')

    % Set strings to MemoryType, DataType, Zone and ContactType
    % Makes arrays (supposedly) faster to check than categorical
    out.MemoryType  = string(out.MemoryType);
    out.DataType    = string(out.DataType);
    out.Zone        = string(out.Zone);
    out.ContactType = string(out.ContactType);
    out.Name        = string(out.Name);

    % Add VarType to speed array checks (saves string comparisons)
    % Calculate VarType according to MemoryType and DataType
    % VarType = 1 Input Bool
    % VarType = 2 Input Float
    % VarType = 3 Input DateTime
    % VarType = 4 Output Bool
    % VarType = 5 Output Float
    % VarType = 6 Output DateTime
    % VarType = 7 Memory Bool
    % VarType = 8 Memory Float
    % VarType = 9 Memory DateTime

    out.VarType = ...
        1 * (out.MemoryType == 'Input' & out.DataType == 'Bool') + ...
        2 * (out.MemoryType == 'Input' & out.DataType == 'Float') + ...
```

```
            3 * (out.MemoryType == 'Input' & out.DataType == 'DateTime') + ...
            4 * (out.MemoryType == 'Output' & out.DataType == 'Bool') + ...
            5 * (out.MemoryType == 'Output' & out.DataType == 'Float') + ...
            6 * (out.MemoryType == 'Output' & out.DataType == 'DateTime') + ...
            7 * (out.MemoryType == 'Memory' & out.DataType == 'Bool') + ...
            8 * (out.MemoryType == 'Memory' & out.DataType == 'Float') + ...
            9 * (out.MemoryType == 'Memory' & out.DataType == 'DateTime');

        out = movevars(out,'VarType','Before','MemoryType');

        % Sort to keep the like variables near one another
        % Mainly affects Outputs, which are really unordered
        out = sortrows(out,{'VarType','Address'},'ascend');

        % Add RowID parameter to make searches faster and less cluttery
        % Calculate RowIDs here if applicable
        for i=1:height(out)
            out.RowID(i) = i;
        end

        out = movevars(out,'RowID','Before','VarType');
    end
```

4.3.4 changePower10V

- **Nombre del método**: changePower10V
- **Visibilidad** (público o privado): privado
- **Entradas requeridas**: (ninguna)
- **Entradas opcionales**: (ninguna)
- **Funcionamiento esperado**: adapta la potencia de los dispositivos *Output* y *Float* que trabajan en el rango de 0-10 V para simplificar posteriores cálculos de potencia. El cambio se hace en el propio atributo *FullData* antes de ser separado en los demás atributos con subconjuntos de estas entradas.
- **Salidas esperadas**: (ninguna, el cambio se hace sobre el propio valor del atributo)
- **Código:**

```
    function changePower10V(obj)
        %changePower10V divides by 10 these outputs that range in the
        %value 0-10V, so that they will consume power proportional to
        %that voltage
        % All of these are Output Floats, VarType == 5

        rows = obj.BaseData.VarType == 5;
        obj.BaseData.Power(rows) = obj.BaseData.Power(rows)/10;
    end
```

4.3.5 splitData

- **Nombre del método**: splitData
- **Visibilidad** (público o privado): privado
- **Entradas requeridas**: (ninguna)
- **Entradas opcionales**: (ninguna)
- **Funcionamiento esperado**: llena los valores de todos los métodos basados en dispositivos de la vivienda de Home I/O, para facilitar su búsqueda y utilización. Se hacen filtros por características siempre que se puede, aunque en el caso del atributo *EnvironmentValues* no ha sido posible y se ha filtrado directamente por direcciones de memoria al estar contiguas.
- **Salidas esperadas**: (ninguna, el cambio se hace sobre los propios valores de los atributos)
- **Código:**

```
        function splitData(obj)
            % splitData splits the data in zones (rooms) and other logical
            % categories, to enable ease of usage by the end user.

            %% Split per useful categories

            % Memory Types
            obj.Types.Inputs   = obj.BaseData(obj.BaseData.MemoryType == 'Input',:);
            obj.Types.Outputs  = obj.BaseData(obj.BaseData.MemoryType == 'Output',:);
            obj.Types.Memories = obj.BaseData(obj.BaseData.MemoryType == 'Memory',:);

            obj.Types.Bools     = obj.BaseData(obj.BaseData.DataType == 'Bool',:);
            obj.Types.Floats    = obj.BaseData(obj.BaseData.DataType == 'Float',:);
            obj.Types.DateTimes = obj.BaseData(obj.BaseData.DataType ==
'DateTime',:);

            obj.Types.InputBools      = obj.BaseData(obj.BaseData.VarType == 1,:);
            obj.Types.InputFloats     = obj.BaseData(obj.BaseData.VarType == 2,:);
            obj.Types.InputDateTimes  = obj.BaseData(obj.BaseData.VarType == 3,:);
            obj.Types.OutputBools     = obj.BaseData(obj.BaseData.VarType == 4,:);
            obj.Types.OutputFloats    = obj.BaseData(obj.BaseData.VarType == 5,:);
            obj.Types.OutputDateTimes = obj.BaseData(obj.BaseData.VarType == 6,:);
            obj.Types.MemoryBools     = obj.BaseData(obj.BaseData.VarType == 7,:);
            obj.Types.MemoryFloats    = obj.BaseData(obj.BaseData.VarType == 8,:);
            obj.Types.MemoryDateTimes = obj.BaseData(obj.BaseData.VarType == 9,:);

            % Zones
            obj.Zones.ZoneA    = obj.BaseData(obj.BaseData.Zone == 'A',:);
            obj.Zones.ZoneB    = obj.BaseData(obj.BaseData.Zone == 'B',:);
            obj.Zones.ZoneC    = obj.BaseData(obj.BaseData.Zone == 'C',:);
            obj.Zones.ZoneD    = obj.BaseData(obj.BaseData.Zone == 'D',:);
            obj.Zones.ZoneE    = obj.BaseData(obj.BaseData.Zone == 'E',:);
            obj.Zones.ZoneF    = obj.BaseData(obj.BaseData.Zone == 'F',:);
            obj.Zones.ZoneG    = obj.BaseData(obj.BaseData.Zone == 'G',:);
            obj.Zones.ZoneH    = obj.BaseData(obj.BaseData.Zone == 'H',:);
            obj.Zones.ZoneI    = obj.BaseData(obj.BaseData.Zone == 'I',:);
            obj.Zones.ZoneJ    = obj.BaseData(obj.BaseData.Zone == 'J',:);
            obj.Zones.ZoneK    = obj.BaseData(obj.BaseData.Zone == 'K',:);
            obj.Zones.ZoneL    = obj.BaseData(obj.BaseData.Zone == 'L',:);
            obj.Zones.ZoneM    = obj.BaseData(obj.BaseData.Zone == 'M',:);
            obj.Zones.ZoneN    = obj.BaseData(obj.BaseData.Zone == 'N',:);
            obj.Zones.ZoneO    = obj.BaseData(obj.BaseData.Zone == 'O',:);
            obj.Zones.ZoneNone = obj.BaseData(obj.BaseData.Zone == '-',:);
```

```
            % By object types
            % Input Bools
            obj.Devices.LightSwitches         =
obj.Types.InputBools(contains(obj.Types.InputBools.Name,"Light Switch") &
~contains(obj.Types.InputBools.Name,"Dimmer"),:);
            obj.Devices.UpDownSwitches        =
obj.Types.InputBools(contains(obj.Types.InputBools.Name,"Up/Down Switch"),:);
            obj.Devices.LightDimmersUpDown    =
obj.Types.InputBools(contains(obj.Types.InputBools.Name,"Light Switch Dimmer"),:);
            obj.Devices.DoorDetectors         =
obj.Types.InputBools(contains(obj.Types.InputBools.Name,"Door Detector"),:);
            obj.Devices.MotionDetectors       =
obj.Types.InputBools(contains(obj.Types.InputBools.Name,"Motion Detector"),:);
            obj.Devices.BrightnessSensorsBool =
obj.Types.InputBools(contains(obj.Types.InputBools.Name,"Brightness Sensor"),:);
            obj.Devices.SmokeDetectors        =
obj.Types.InputBools(contains(obj.Types.InputBools.Name,"Smoke Detector"),:);
            obj.Devices.AlarmKeyPadArmed      =
obj.Types.InputBools(contains(obj.Types.InputBools.Name,"Alarm Key Pad"),:);
            obj.Devices.GarageDoorSensors     =
obj.Types.InputBools(contains(obj.Types.InputBools.Name,"Garage Door"),:);
            obj.Devices.EntranceGateSensors   =
obj.Types.InputBools(contains(obj.Types.InputBools.Name,"Entrance Gate"),:);
            obj.Devices.RemoteButtons         =
obj.Types.InputBools(contains(obj.Types.InputBools.Name,"Remote Button"),:);

            % Input Floats
            obj.Devices.BrightnessSensorsFloat =
obj.Types.InputFloats(contains(obj.Types.InputFloats.Name,"Brightness Sensor"),:);
            obj.Devices.ThermostatTemperatures =
obj.Types.InputFloats(contains(obj.Types.InputFloats.Name,"(Room Temperature)"),:);
            obj.Devices.ThermostatSetPoints    =
obj.Types.InputFloats(contains(obj.Types.InputFloats.Name,"(Set Point)"),:);
            obj.Devices.RollerShades           =
obj.Types.InputFloats(contains(obj.Types.InputFloats.Name,"Roller Shades"),:);

            % Input DateTime
            obj.Devices.DateAndTimeInput =
obj.Types.InputDateTimes(contains(obj.Types.InputDateTimes.Name,"Date and Time"),:);

            % Output Bools
            obj.Devices.LightsBool          =
obj.Types.OutputBools(contains(obj.Types.OutputBools.Name,"Lights"),:);
            obj.Devices.RollerShadesUpDown  =
obj.Types.OutputBools(contains(obj.Types.OutputBools.Name,"Roller Shades"),:);
            obj.Devices.HeatersBool         =
obj.Types.OutputBools(contains(obj.Types.OutputBools.Name,"Heater"),:);
            obj.Devices.Sirens              =
obj.Types.OutputBools(contains(obj.Types.OutputBools.Name,"Siren"),:);
            obj.Devices.AlarmKeyPad         =
obj.Types.OutputBools(contains(obj.Types.OutputBools.Name,"Alarm Key Pad"),:);
            obj.Devices.GarageDoorOpenClose =
obj.Types.OutputBools(contains(obj.Types.OutputBools.Name,"Garage Door"),:);
            obj.Devices.EntranceGate        =
obj.Types.OutputBools(contains(obj.Types.OutputBools.Name,"Entrance Gate"),:);

            % Output Floats
            obj.Devices.LightsFloat  =
obj.Types.OutputFloats(contains(obj.Types.OutputFloats.Name,"Lights"),:);
```

```
            obj.Devices.HeatersFloat =
obj.Types.OutputFloats(contains(obj.Types.OutputFloats.Name,"Heater"),:);

            % Output DateTimes
            % Nonexistent

            % Memory Bools
            obj.Devices.DLST =
obj.Types.MemoryBools(contains(obj.Types.MemoryBools.Name,"DLST"),:);

            % Memory Floats
            obj.Devices.TimeScale         =
obj.Types.MemoryFloats(contains(obj.Types.MemoryFloats.Name,"Time Scale"),:);
            obj.Devices.LatLongitude      =
obj.Types.MemoryFloats(contains(obj.Types.MemoryFloats.Name,["Latitude",
"Longitude"]),:);
            obj.Devices.EnvironmentValues =
obj.Types.MemoryFloats(obj.Types.MemoryFloats.Address >= 132 &
obj.Types.MemoryFloats.Address <= 140,:);
            obj.Devices.InstantPower      =
obj.Types.MemoryFloats(contains(obj.Types.MemoryFloats.Name,"Instant Power"),:);
            obj.Devices.CurrentPower      =
obj.Types.MemoryFloats(contains(obj.Types.MemoryFloats.Name,"Current"),:);
            obj.Devices.LastPower         =
obj.Types.MemoryFloats(contains(obj.Types.MemoryFloats.Name,"Last"),:);
            obj.Devices.ZoneTemperatures  =
obj.Types.MemoryFloats(obj.Types.MemoryFloats.Zone ~= "-",:);

            % Memory DateTimes
            obj.Devices.DateAndTimeMemory =
obj.Types.MemoryDateTimes(contains(obj.Types.MemoryDateTimes.Name,"Date"),:);


            % Other memberships
            obj.Special.InputsNO        =
obj.Types.InputBools(obj.Types.InputBools.ContactType == 'NO',:);
            obj.Special.InputsNC        =
obj.Types.InputBools(obj.Types.InputBools.ContactType == 'NC',:);
            obj.Special.Inputs10V       = [obj.Devices.BrightnessSensorsFloat;
obj.Devices.RollerShades];
            obj.Special.Outputs10V      = [obj.Devices.LightsFloat;
obj.Devices.HeatersFloat];

            obj.Special.ConflictInputs   = [obj.Devices.UpDownSwitches;
obj.Devices.LightDimmersUpDown];
            obj.Special.ConflictOutputs  = [obj.Devices.RollerShadesUpDown;
obj.Devices.GarageDoorOpenClose; obj.Devices.EntranceGate; obj.Devices.AlarmKeyPad];

            % Special.BoolFloatOutputs will save us time if stored
            % alternating the corresponding Bool and Float consecutively
            % Not pretty but useful. TOO SLOW but only runs once
            obj.Special.BoolFloatOutputs = table();

            for i=1:size(obj.Devices.LightsBool)
                obj.Special.BoolFloatOutputs = [obj.Special.BoolFloatOutputs;
obj.Devices.LightsBool(i,:); obj.Devices.LightsFloat(i,:)];
            end

            for i=1:size(obj.Devices.HeatersBool)
                obj.Special.BoolFloatOutputs = [obj.Special.BoolFloatOutputs;
obj.Devices.HeatersBool(i,:); obj.Devices.HeatersFloat(i,:)];
```

```
            end

        end
```

4.3.6 checkHomeIO

- **Nombre del método**: checkHomeIO
- **Visibilidad** (público o privado): público
- **Entradas requeridas**: (ninguna)
- **Entradas opcionales**: un argumento dado por nombre y valor:
    - *action*: texto que define la acción que se debería tomar en caso de que no se encuentre de forma adecuada el proceso *Home I/O*. Debe ser '*warning'* (lanza un warning), '*error'* (lanza un error), o '*none'* (no lanza nada). Su valor por defecto es '*none'*.
- **Funcionamiento esperado**: Si no se encuentra únicamente un proceso llamado "*Home IO*", devuelve el comportamiento definido según la acción *action*.
- **Salidas esperadas**:
    - ninguna si hay un único proceso "*Home IO*" en ejecución, o *action* es 'none'.
    - un warning en otro caso, si *action* es *'warning'*.
    - un error en otro caso, si *action* es *'error'*, pasando a eliminar el objeto *HomeIO* para evitar problemas adicionales.
- **Código:**

```
        function out = checkHomeIO(obj,varargin)
            %checkHomeIO checks if Home I/O is running and warns or errors
            %   if appropriate value is supplied

            %% Parse inputs
            default_action = 'none';

            p = inputParser;
            p.addParameter('action',default_action,@mustBeText);
            p.parse(varargin{:});

            action = p.Results.action;
            mustBeMember(action,{'warning','error','none'});

            out = System.Diagnostics.Process.GetProcessesByName("Home IO").Length;

            if action ~= "none" && out ~=1
                if action == "warning"
                    warning('Unable to find Home I/O process or Home I/O running more
than once. Results might be inaccurate.')
                    out = 0;
                elseif action == "error"
                    obj.delete()
                    error('Unable to find Home I/O process or Home I/O running more
than once. Stopping...')
                end
            end
        end
```

4.3.7 updateHomeIO

- **Nombre del método**: updateHomeIO
- **Visibilidad** (público o privado): público
- **Entradas requeridas**: (ninguna)
- **Entradas opcionales**: (ninguna)
- **Funcionamiento esperado**: Simplemente llama al método *EngineIO.MemoryMap.Instance.Update()* de *Home I/O* para refrescar la copia de memoria en caché. Si los escuchadores de eventos se encuentran activos por la configuración inicial del objeto *HomeIO*, los respectivos eventos de datos cambiados saltarán automáticamente al actualizarse la copia en caché. Método estático.
- **Salidas esperadas**: (ninguna)
- **Código:**

```
function updateHomeIO()
    %Simply runs the Home I/O command to update the MemoryMap
    EngineIO.MemoryMap.Instance.Update();
end
```

4.3.8 readValues

- **Nombre del método**: readValues
- **Visibilidad** (público o privado): público
- **Entradas requeridas**: (ninguna)
- **Entradas opcionales**:
    - *data*: subconjunto de datos de los dispositivos sobre los que leer. Este valor *data* debe contener miembros completos de *BaseData*. En caso de no darse, su por defecto es el conjunto *BaseData* por completo.
- **Funcionamiento esperado**: accede al *SDK* de *Home I/O* para leer los valores especificados en *data* desde la caché de memoria de intercambio con el simulador de vivienda. Los valores *DateTime* son almacenados en el formato *datenum* de *MATLAB*. Si *data* está vacío (o es *BaseData*), se toman todos los valores de todos los registros de *Home I/O*. Esta función sin argumento se suele utilizar para actualizar completamente los valores de *FullData.*
- **Salidas esperadas**: devuelve los valores (y únicamente los valores) del conjunto *values* recién leídos de *Home I/O*.
- **Código**

```
function values = readValues(obj,varargin)
    %Reads values from the supplied array in the argument. If no
    %   argument is provided, reads memories from every line
    %   included in the base data property (obj.BaseData)

    %% Parse inputs
    default_data = obj.BaseData;

    p = inputParser;
    p.addOptional('data',default_data); % Must be member of BaseData
    p.parse(varargin{:});

    data = p.Results.data;
    obj.mustMember(data,obj.BaseData);
```

```
            %% Work with data

            % Preset values with NaN of correct size
            values = nan(height(data),1);

            % Attempt at speeding up more: switch-case method
            for i=1:height(data)
                switch data.VarType(i)
                    case 1
                        values(i) =
EngineIO.MemoryMap.Instance.GetBit(data.Address(i), EngineIO.MemoryType.Input).Value;
                    case 2
                        values(i) =
EngineIO.MemoryMap.Instance.GetFloat(data.Address(i),
EngineIO.MemoryType.Input).Value;
                    case 3
                        % Datenum accepts double values only
                        temp =
EngineIO.MemoryMap.Instance.GetDateTime(data.Address(i),
EngineIO.MemoryType.Input).Value;
                        temp = double([temp.Year temp.Month temp.Day temp.Hour
temp.Minute temp.Second+temp.Millisecond/1000]);
                        values(i) = datenum(temp(1), temp(2), temp(3), temp(4),
temp(5), temp(6));

                    case 4
                        values(i) =
EngineIO.MemoryMap.Instance.GetBit(data.Address(i),
EngineIO.MemoryType.Output).Value;
                    case 5
                        values(i) =
EngineIO.MemoryMap.Instance.GetFloat(data.Address(i),
EngineIO.MemoryType.Output).Value;
                    case 6 % Does never happen but to keep code consistency
                        % Datenum accepts double values only
                        temp =
EngineIO.MemoryMap.Instance.GetDateTime(data.Address(i),
EngineIO.MemoryType.Output).Value;
                        temp = double([temp.Year temp.Month temp.Day temp.Hour
temp.Minute temp.Second+temp.Millisecond/1000]);
                        values(i) = datenum(temp(1), temp(2), temp(3), temp(4),
temp(5), temp(6));

                    case 7
                        values(i) =
EngineIO.MemoryMap.Instance.GetBit(data.Address(i),
EngineIO.MemoryType.Memory).Value;
                    case 8
                        values(i) =
EngineIO.MemoryMap.Instance.GetFloat(data.Address(i),
EngineIO.MemoryType.Memory).Value;
                    case 9
                        % Datenum accepts double values only
                        temp =
EngineIO.MemoryMap.Instance.GetDateTime(data.Address(i),
EngineIO.MemoryType.Memory).Value;
                        temp = double([temp.Year temp.Month temp.Day temp.Hour
temp.Minute temp.Second+temp.Millisecond/1000]);
```

```
                        values(i) = datenum(temp(1), temp(2), temp(3), temp(4),
temp(5), temp(6));
                    otherwise
                        error("Unrecognized case: got an unknown value at data row
%i",i);
                end
            end
            % returns values
        end
```

4.3.9 onTimerCall

- **Nombre del método**: onTimerCall
- **Visibilidad** (público o privado): público
- **Entradas requeridas**: (ninguna)
- **Entradas opcionales**: (ninguna)
- **Funcionamiento esperado**: esta función está pensada para ejecutarse cada vez a cada disparo del temporizador, si los escuchadores de eventos se encuentran activos. Comprueba que el programa Home I/O sigue en funcionamiento (llamando a *checkHomeIO*) y actualiza la copia en cache (llamando a *updateHomeIO*). Los escuchadores de eventos disparan a su vez sus funciones si se da el evento de cambio de valores en el tipo de registro en el que se encuentran a la escucha. La visibilidad de este método es pública, por si se da el caso de necesitar lanzarlo manualmente sin el temporizador.
- **Salidas esperadas**: ninguna, aunque se espera el disparo de los eventos en caso de que haya un cambio en valores de Home I/O.
- **Código:**

```
        function OnTimerCall(obj,~,~)
            %Timer callback if event handlers are available

            %profile resume
            %tim = toc;
            obj.checkHomeIO("action","error");
            obj.updateHomeIO(); % Callback is automatically executed by listener
handles
            %disp(toc-tim);
            %profile off
        end
```

4.3.10 onTimerCallNoListener

- **Nombre del método**: onTimerCallNoListener
- **Visibilidad** (público o privado): público
- **Entradas requeridas**: (ninguna)
- **Entradas opcionales**: (ninguna)
- **Funcionamiento esperado**: esta función es similar a la anterior (*OnTimerCall*), en caso en que los escuchadores no se encuentren activos. En este caso, además de comprobar la ejecución de *Home I/O* y actualizar sus valores, se llama a *readValues* para hacer una actualización indiscriminada de los valores de todos los dispositivos de la casa en el atributo *FullData*. Su visibilidad también es pública para el caso en que se necesite disparar manualmente.

- **Salidas esperadas**: (ninguna, los valores devueltos por readValues actualizan al atributo *FullData*).
- **Código:**

```
function OnTimerCallNoListener(obj,~,~)
    %Timer Callback if listener handles are deactivated. Update is
    %   done manually via the readValues method.

    %profile resume
    %tim = toc;
    obj.checkHomeIO("action","error");
    obj.updateHomeIO();
    obj.FullData.Value = obj.readValues();
    %disp(toc-tim);
    %profile off
end
```

4.3.11 getRows

- **Nombre del método**: getRows
- **Visibilidad** (público o privado): público
- **Entradas requeridas**: (ninguna)
- **Entradas opcionales**:
    - *data*: subconjunto de datos de los dispositivos sobre los que leer. Este valor *data* debe contener miembros completos (*filas*) del atributo *BaseData* (o al menos una columna *RowID* con los mismos identificadores que en *BaseData*, en otro caso devolverá error). En caso de no darse, su por defecto es el conjunto *BaseData* por completo.
- **Funcionamiento esperado**: devuelve filas completas del atributo *FullData* (con valores), siendo *data* filas completas (o que contengan al menos una columna *RowID* con los mismos identificadores que en *BaseData*) del atributo *BaseData*. Se llama a *getRowsFromRowIDs* para mayor velocidad de búsqueda en comparación a encontrar miembros completos de *data* que coincidan con filas completas del atributo *BaseData*. Si alguna fila de *data* no se encuentra en el atributo *BaseData*, devolverá error.
- **Salidas esperadas**: filas completas del atributo *FullData*, que contengan los datos de *data*.
- **Código:**

```
function out = getRows(obj,varargin)
    %getRows fetches appropriate rows from the obj.FullData
    %   property. If no data array is supplied, returns the full
    %   table of obj.FullData

    %% Parse inputs
    default_data = obj.BaseData;

    p = inputParser;
    p.addOptional('data',default_data); % Must be member of BaseData
    p.parse(varargin{:});

    data = p.Results.data;
    mustBeMember(data,obj.BaseData);

    %% Select and return rows from obj.FullData
    out = obj.getRowsFromRowIDs(data.RowID);

end
```

4.3.12 getRowsFromRowIDs

- **Nombre del método**: getRowsFromRowIDs
- **Visibilidad** (público o privado): público
- **Entradas requeridas**:
  - *data*: vector columna que contiene aquellos *identificadores de filas* de los dispositivos que leer del atributo *FullData*.
- **Entradas opcionales**: (ninguna)
- **Funcionamiento esperado**: devuelve las filas completas de *FullData,* identificadas de forma única a través de los identificadores de fila provistos en *data*. Si un valor de *data* se encuentra fuera del rango de identificadores de filas contenido en el atributo *FullData*, devolverá error.
- **Salidas esperadas**: filas completas del atributo *FullData* identificadas por *data*.
- **Código:**

```
function out = getRowsFromRowIDs(obj,rows)
    p = inputParser;
    p.addRequired('rows',@mustBeNumeric);
    p.parse(rows);

    out = obj.FullData(rows,:);
end
```

4.3.13 getValues

- **Nombre del método**: getValues
- **Visibilidad** (público o privado): público
- **Entradas requeridas**: (ninguna)
- **Entradas opcionales**:
  - *data*: subconjunto de datos de los dispositivos sobre los que leer. Este valor *data* debe contener miembros completos (filas) del atributo *BaseData* (o al menos una columna *RowID* con los mismos identificadores que en *BaseData*, en otro caso devolverá error). En caso de no darse, su por defecto es el conjunto *BaseData* por completo.
- **Funcionamiento esperado**: devuelve los valores (y únicamente los valores) registrados en el atributo *FullData*, siendo *data* filas completas del atributo *BaseData* (o que contengan al menos una columna *RowID* con los mismos identificadores que en *BaseData*). Se llama a *getValuesFromRowIDs* para mayor velocidad de búsqueda en comparación a encontrar miembros completos de *data* que coincidan con filas completas del atributo *BaseData*. Si alguna fila de *data* no se encuentra en el atributo *BaseData*, devolverá error.
- **Salidas esperadas**: Columna *Value* (valores) del atributo *FullData*, que contengan los datos de *data*.
- **Código:**

```
function out = getValues(obj,varargin)
    %getValues fetches current values (only) from the rows of the
    %   obj.FullData property. If no data array is supplied,
    %   returns the full values table of obj.FullData

    %% Parse inputs
    default_data = obj.BaseData;
```

```
            p = inputParser;
            p.addOptional('data',default_data); % Must be member of BaseData
            p.parse(varargin{:});

            data = p.Results.data;
            mustBeMember(data,obj.BaseData);

            %% Select and return values from obj.FullData
            out = obj.getValuesFromRowIDs(data.RowID);

        end
```

4.3.14 getValuesFromRowIDs

- **Nombre del método**: getValuesFromRowIDs
- **Visibilidad** (público o privado): público
- **Entradas requeridas**:
    - *data*: vector columna que contiene aquellos *identificadores de filas* de los dispositivos que leer del atributo *FullData*.
- **Entradas opcionales**: (ninguna)
- **Funcionamiento esperado**: devuelve los valores (y únicamente los valores) registrados en el atributo *FullData,* identificados de forma única a través de los identificadores de fila provistos en *data*. Si un valor de *data* se encuentra fuera del rango de identificadores de filas contenido en el atributo *FullData*, devolverá error.
- **Salidas esperadas**: Columna *Value* (valores) del atributo *FullData* identificadas por *data*.
- **Código:**

```
        function out = getValuesFromRowIDs(obj,rows)
            p = inputParser;
            p.addRequired('rows',@mustBeNumeric);
            p.parse(rows);

            out = obj.FullData.Value(rows);
        end
```

4.3.15 estimatePower

- **Nombre del método**: estimatePower
- **Visibilidad** (público o privado): público
- **Entradas requeridas**: (ninguna)
- **Entradas opcionales**: (ninguna)
- **Funcionamiento esperado**: calcula una estimación del valor de la potencia consumida en la vivienda de *Home I/O* (los valores de consumos flotantes en el rango 0-10V deben haber sido adaptados con el método *changePower10V* en la construcción del objeto HomeIO) a través de multiplicar el consumo de los dispositivos por su valor de funcionamiento. En caso de que haya valores de dispositivos *Output* escritos en conflicto (encendidos a la vez en modo binario y flotante), la estimación será errónea. Estos conflictos deben ser subsanados de forma externa.
- **Salidas esperadas**: el valor estimado de la potencia, en W.

- **Código**

```
function out = estimatePower(obj)
    %estimatePower returns the power consumption estimated via the
    % devices currently active. Does not need Home IO running.
    out = sum(obj.FullData.Power.*obj.FullData.Value);
end
```

4.3.16 setValues

- **Nombre del método**: setValues
- **Visibilidad** (público o privado): público
- **Entradas requeridas**:
    - *data*: subconjunto de datos de los dispositivos sobre los que leer. Este valor *data* debe contener miembros completos (filas) del atributo *BaseData* (o al menos una columna *RowID,* una columna *DataType* de tipo válido y una columna *Address* con una dirección de memoria válida) con los mismos identificadores que en *BaseData*, en otro caso devolverá error.
    - *values:* vector columna que contiene los datos que se van a dar a los dispositivos identificados por *data,* manteniendo el mismo orden y con la misma longitud (número de filas) que las filas dadas en *data*. El vector debe ser numérico o la función producirá un error.
- **Entradas opcionales**:
    - *checkHomeIO*: valor booleano. Se establecerá en *Verdadero* si se quiere comprobar si *Home I/O* se encuentra en funcionamiento antes de hacer el cambio de valor (llamando al método *checkHomeIO* con un *action* de *'error'*). En caso *Verdadero*, se producirá un error si el simulador de vivienda no está en ejecución (y se encuentra ejecutando una única vez). Por defecto es *Verdadero*.
    - *capValues*: valor booleano. Se establecerá en *Verdadero* si se quiere limitar los valores de las salidas, de modo que un valor binario tenga valores únicamente de 0 o 1 o los valores flotantes limitados en el rango 0-10 sean limitados a este rango. Por defecto es *Verdadero*.
    - *checkConflicts*: valor booleano. Se establecerá en *Verdadero* si se quiere comprobar conflictos entre valores de objetos que sean dispositivos que pueden operar a la vez en régimen binario y flotante. Si es *Verdadero*, la función buscará el otro modo de operación del dispositivo y lo pondrá en valor 0 (lo desactivará). Por defecto es *Verdadero.*
    - *update*: valor booleano. Se establecerá en *Verdadero* si se quiere actualizar *Home I/O* tras la operación de asignación (llamando al método *updateHomeIO*), de modo que los cambios que se hagan tomen efecto inmediato en el simulador. Por defecto es *Verdadero*.
- **Funcionamiento esperado**: El método comprueba los valores aportados: que *data* sean filas tipo *Output* del atributo *BaseData* y que *values* sea numérico, además de que concuerdan en longitud. Posteriormente, si *checkHomeIO* es *Verdadero*, comprueba que *Home I/O* está en ejecución. Después, comprueba que cada valor está en rango (si *capValues* es *Verdadero*) y que no se producen conflictos (si *checkConflicts* es *Verdadero*; en caso de que se produzcan conflictos desactiva la entrada conflictiva). Una vez hechas estas comprobaciones, produce los cambios en las filas aportadas en *data* con los valores aportados en *values*. Si *update* es *Verdadero*, llama a *updateHomeIO* para que los cambios tengan efecto en la vivienda simulada.
- **Salidas esperadas**: (ninguna, los cambios toman efecto inmediato)
- **Código:**

```
        function setValues(obj,data,values,varargin)
            %setValues sets values to specified data. Only works for Output
            %   values as these are the only settable (gives a warning
            %   otherwise).
            %   Required parameters: data and values;
            %   Optional parameters: CheckConflicts deactivates conflicting
            %   outputs (for example open and close garage door, keeps the
            %   last one); CapValues caps Bool values to integer 0-1 and
            %   Float values that have 0-10 limits

            %% Parse inputs

            def_conflicts   = true;
            def_capvalues   = true;
            def_update      = true;
            def_checkHomeIO = true;

            p = inputParser;
            p.addRequired("data")  % Unable to find verif. function
            p.addRequired("values") % Same
            p.addParameter("checkConflicts",def_conflicts,@mustBeNumericOrLogical)
            p.addParameter("capValues",def_capvalues,@mustBeNumericOrLogical)
            p.addParameter("update",def_update,@mustBeNumericOrLogical)
            p.addParameter("checkHomeIO",def_checkHomeIO,@mustBeNumericOrLogical)

            p.parse(data,values,varargin{:});

            data           = p.Results.data;
            values         = p.Results.values;
            checkConflicts = p.Results.checkConflicts;
            capValues      = p.Results.capValues;
            update         = p.Results.update;
            checkHomeIO    = p.Results.checkHomeIO;

            %% Check HomeIO since this petition can come from outside,
            % also check and sanitize other variable issues
            if checkHomeIO
                obj.checkHomeIO("action","error");
            end

            obj.mustMember(data,obj.Types.Outputs);
            mustBeNumericOrLogical(values); % NaNs get hunted later

            % Reject if no coincidence in heights
            n = height(data);
            if height(values) ~= n
                error("Height of data (%d) does not match with height of values
provided (%d)",n,height(values))
            end

            %% Process values
            % Prepare checks for membership for all the data array in a
            % timely manner
            if capValues
                % Just a logical value works
                capRows = obj.checkMember(data,obj.Special.Outputs10V);
            end
            if checkConflicts
                % We need the row value here
                [~,conflictRows] = obj.checkMember(data,obj.Special.ConflictOutputs);
```

```
                [~,BoolFloatRows] =
obj.checkMember(data,obj.Special.BoolFloatOutputs);
            end

            % Get the home devices and set the values in different steps.

            for i=1:n
                % Get device type and address to temp var (no table)
                % Addresses must be extracted from the table too
                type = data.DataType(i); % "Bool", "Float", no "DateTime"
                addr = data.Address(i);

                if type == "Bool"
                    device = EngineIO.MemoryMap.Instance.GetBit(addr,
EngineIO.MemoryType.Output);
                elseif type == "Float"
                    device = EngineIO.MemoryMap.Instance.GetFloat(addr,
EngineIO.MemoryType.Output);
                end

                % Use the capValues and checkConflicts if asked
                if capValues
                    if type == "Bool"
                        if values(i)
                            values(i) = 1;
                        else
                            values(i) = 0;
                        end
                    elseif type == "Float"
                        if capRows(i)
                            if values(i) > 10
                                values(i) = 10;
                            elseif values(i) < 0
                                values(i) = 0;
                                %  else % Not needed
                                %  values(i) = values(i);
                            end
                        end
                    else % There are no output DateTimes, throw an error
                        error("Unrecognized output type %s for data address
%d",type,addr);
                    end
                end

                % Odd row: Disable next value
                % Even row: Disable prev. value
                if checkConflicts
                    % Incompatible bool values
                    if conflictRows(i)
                        if rem(conflictRows(i),2) % == 1
                            addr2 =
obj.Special.ConflictOutputs.Address(conflictRows(i)+1);
                            %addr2 = addr2{:,:};
                            device2 = EngineIO.MemoryMap.Instance.GetBit(addr2,
EngineIO.MemoryType.Output);
                        else % rem(conflictRows(i),2) == 0
                            addr2 =
obj.Special.ConflictOutputs.Address(conflictRows(i)-1);
                            %addr2 = addr2{:,:};
                            device2 = EngineIO.MemoryMap.Instance.GetBit(addr2,
EngineIO.MemoryType.Output);
```

```
                        end
                        device2.Value = 0;
                    end

                    % Incompatible Bool/Float values
                    if BoolFloatRows(i)
                        if rem(BoolFloatRows(i),2) % == 1, it's a Bool so the other
one is Float
                            addr2 =
obj.Special.BoolFloatOutputs.Address(BoolFloatRows(i)+1);
                            device2 = EngineIO.MemoryMap.Instance.GetFloat(addr2,
EngineIO.MemoryType.Output);
                        else % rem(idx,2) == 0, , it's a Float so the other one is
Bool
                            addr2 =
obj.Special.BoolFloatOutputs.Address(BoolFloatRows(i)-1);
                            device2 = EngineIO.MemoryMap.Instance.GetBit(addr2,
EngineIO.MemoryType.Output);
                        end
                        device2.Value = 0;
                    end

                end

                % Set all values now
                device.Value = values(i);
            end

            % Sometimes all other values update correctly except for these
            % Give our model a hand:
            obj.FullData.Value(data.RowID) = values;

            %% Force update if required; values will be read by the events
            if update
                obj.updateHomeIO();
            end
        end
```

4.3.17 onValuesChanged

- **Nombre del método**: onValuesChanged
- **Visibilidad** (público o privado): privado
- **Entradas requeridas**: (proporcionadas por los manejadores de eventos)
- **Entradas opcionales**: (proporcionadas por los manejadores de eventos)
- **Funcionamiento esperado**: Función que ejecutan los manejadores de eventos (si se encuentran activos). Simplemente se limita a llamar a la función *updateFromEvent* con las listas de valores cambiados, para así procesarlos y actualizar el atributo *FullData*.
- **Salidas esperadas**: (ninguna)
- **Código:**

```
        function OnValuesChanged(obj,varargin)
            % Callback function called by event handlers after updating
            % Home I/O from the update function.

            % varargin{1} is the name of the event, which is useless
            % varargin{2}.MemoriesBit, varargin{2}.MemoriesFloat,
varargin{2}.MemoriesDateTime
            % These three are vectors, length specified in
varargin{2}.Memories[[TYPE]].Length
            % Direct access, eg, by using varargin(2).MemoriesFloat(i)

            obj.updateFromEvent(varargin{2});
        end
```

4.3.18 updateFromEvent

- **Nombre del método**: updateFromEvent
- **Visibilidad** (público o privado): privado
- **Entradas requeridas**:
    - *data*: datos cambiados desde la última actualización (proporcionados automáticamente por los escuchadores de eventos).
- **Entradas opcionales**: (ninguna)
- **Funcionamiento esperado**: A cada valor cambiado se le da un tratamiento según el tipo de dato (*Bit, Float* o *DateTime,* que es el formato con el que viene *data* en el estado en que los envían los escuchadores de eventos). A todos los valores se les busca la entrada en el atributo *FullData* a partir de su tipo de dato, su tipo de registro y su dirección, y se almacena su valor. En caso de que el tipo de dato sea un *DateTime*, antes de almacenar su valor en el atributo *FullData* se convierte su contenido al formato *datenum*, para que quepa en una columna numérica.
- **Salidas esperadas**: (ninguna, la actualización del atributo *FullData* se hace dentro del propio método)
- **Código:**

```
        function updateFromEvent(obj,data)
            %updateFromEvent updates values from the obj.FullData attribute
            %   caught from the events.

            % Get MemoriesBit values
            if data.MemoriesBit.Length > 0
                for i=1:data.MemoriesBit.Length
                    % Get data, then get the index
                    datarow = data.MemoriesBit(i);

                    % Below is equivalent to these 2 lines, but faster
                    %rowid = obj.Types.Bools.RowID( ...
                    %    obj.Types.Bools.MemoryType == string(datarow.MemoryType) &
...
                    %    obj.Types.Bools.Address == datarow.Address);
                    %obj.FullData.Value(rowid) = datarow.Value;

                    obj.FullData.Value( obj.Types.Bools.RowID( ...
                        obj.Types.Bools.MemoryType == string(datarow.MemoryType) &
...
                        obj.Types.Bools.Address == datarow.Address) ...
                        ) = datarow.Value;
```

```
            end
        end

        if data.MemoriesFloat.Length > 0
            for i=1:data.MemoriesFloat.Length
                % Get data, then get the index
                datarow = data.MemoriesFloat(i);

                % Below is equivalent to these 2 lines, but faster
                %rowid = obj.Types.Floats.RowID( ...
                %    obj.Types.Floats.MemoryType == string(datarow.MemoryType) &
...
                %    obj.Types.Floats.Address == datarow.Address);
                %obj.FullData.Value(rowid) = datarow.Value;

                obj.FullData.Value(obj.Types.Floats.RowID( ...
                    obj.Types.Floats.MemoryType == string(datarow.MemoryType) &
...
                    obj.Types.Floats.Address == datarow.Address) ...
                    ) = datarow.Value;

            end
        end

        if data.MemoriesDateTime.Length > 0
            for i=1:data.MemoriesDateTime.Length
                % Get data, then get the index
                datarow = data.MemoriesDateTime(i);

                % Datenum accepts double values only
                temp = datarow.Value;
                temp = double([temp.Year temp.Month temp.Day temp.Hour
temp.Minute temp.Second+temp.Millisecond/1000]);

                obj.FullData.Value(obj.Types.DateTimes.RowID( ...
                    obj.Types.DateTimes.MemoryType == string(datarow.MemoryType)
& ...
                    obj.Types.DateTimes.Address == datarow.Address) ...
                    ) = datenum(temp(1), temp(2), temp(3), temp(4), temp(5),
temp(6));
            end
        end
    end
```

4.3.19 checkMember

- **Nombre del método**: checkMember
- **Visibilidad** (público o privado): público
- **Entradas requeridas**: (idénticas a la función de *MATLAB ismember*)
    - *A*: tabla de la que se quiere comprobar si sus miembros (filas) aparecen también como miembros en la matriz B.
    - *B*: tabla que contiene los valores sobre los que si se quiere comprobar membresía desde A.
- **Entradas opcionales**: (ninguna)

- **Funcionamiento esperado**: Simplemente llama a la función *ismember* que ya se encuentra en *MATLAB*, pasando los valores *RowID* de las tablas aportadas, de modo que la comprobación de membresía entre las tablas A y B es más rápida que realizar una comparación a través de todas las columnas de los valores, puesto que comparar textos resulta mucho más lento que comparar valores numéricos. Método estático que no necesita hacer uso de ningún atributo ni método del propio objeto.
- **Salidas esperadas**: (idénticas a la función de *MATLAB ismember*)
    - *Lia*: vector columna de valores de tipo lógico (booleano), que indica si la fila de la tabla *A* aparece en la tabla *B*. *Verdadero* en caso afirmativo, *Falso* en otro caso.
    - *Locb*: vector columna que indica, en cada una de sus filas, la fila de la tabla *B* en la que se ha encontrado cada fila de la tabla *A*. En caso de que la fila de *A* no esté en *B*, el valor es cero.
- **Código:**

```
function [Lia, Locb] = checkMember(A,B)
    % A wrapper function of ismember for our table properties
    % which speeds up comparisons based on RowIDs instead of
    % checking the full rows and saves a lot of time
    [Lia, Locb] = ismember(A.RowID,B.RowID);
end
```

4.3.20 mustMember

- **Nombre del método**: mustMember
- **Visibilidad** (público o privado): público
- **Entradas requeridas**: (las mismas que la función *mustBeMember* de *MATLAB*)
    - *A*: tabla de la que se quiere comprobar si sus miembros (filas) aparecen también como miembros en la matriz B.
    - *B*: tabla que contiene los valores sobre los que si se quiere comprobar membresía desde A.
- **Entradas opcionales**: (ninguna)
- **Funcionamiento esperado**: Simplemente llama a la función *mustBeMember de* MATLAB, pasando los RowIDs de las tablas aportadas, de modo que se acelera la comprobación de membresía de los miembros de la tabla *A* en la tabla *B* al hacer únicamente una comparación numérica y evitar comparaciones de cadenas de texto, que son más lentas. Método estático que no necesita hacer uso de ningún atributo ni método del propio objeto.
- **Salidas esperadas**: Ninguna si todos los miembros de la tabla *A* están contenidos en la tabla *B*, un error en otro caso.
- **Código:**

```
function mustMember(A,B)
    % A wrapper function of mustBeMember for our table properties
    % which speeds up comparisons based on RowIDs instead of
    % checking the full rows and saves a lot of time
    mustBeMember(A.RowID,B.RowID);
end
```

4.3.21 delete

- **Nombre del método**: delete
- **Visibilidad** (público o privado): público
- **Entradas requeridas**: (ninguna)
- **Entradas opcionales**: (ninguna)
- **Funcionamiento esperado**: *delete* es un método que se ejecuta cuando *MATLAB* intenta borrar un objeto, para causar un cierre ordenado de las funciones. En este caso, el cierre ordenado pasa por la parada del temporizador y los escuchadores de eventos (si los hay, respectivamente) y el borrado del mapa de memoria de *Home I/O*. El objeto sigue quedando accesible (comportamiento deseado por *MATLAB*), pero vacío.
- **Salidas esperadas**: (ninguna, pero el objeto debe quedar vacío)
- **Código:**

```
function delete(obj)
    %Remove timers and listeners to avoid unnecessary cluttering
    if obj.Config.set_timer
        stop(obj.Comm.t);
        delete(obj.Comm.t);
    end

    if obj.Config.set_listener
        delete(obj.Comm.lh1);
        delete(obj.Comm.lh2);
        delete(obj.Comm.lh3);
    end
    EngineIO.MemoryMap.Instance.Dispose();
end
```

4.3.22 saveobj

- **Nombre del método**: saveobj
- **Visibilidad** (público o privado): público
- **Entradas requeridas**: (ninguna)
- **Entradas opcionales**: (ninguna)
- **Funcionamiento esperado**: *saveobj* es un método que se ejecuta si *MATLAB* lo encuentra cuando se intenta guardar el objeto en un archivo *.MAT*. En el caso del objeto *HomeIO*, toda la información para reconstruir el objeto a través del método *loadobj* se encuentra en el atributo *Config*, por lo que esa estructura es lo que se guarda. Esto es importante si se intentan guardar escuchadores, porque por sus propiedades son objetos que no se pueden copiar y causarán un *warning* al intentar restaurar este objeto desde un archivo *.MAT*.
- **Salidas esperadas**:
  - *s*: una estructura (*struct*) de MATLAB, con los datos para la reconstrucción del objeto.
- **Código:**

```
function s = saveobj(obj)
    %Save our object to a class to reconstruct it. Configuration
    % parameters are just enough to accomplish this task.
    s.Config = obj.Config;
```

```
        end
```

4.3.23 loadobj

- **Nombre del método**: loadobj
- **Visibilidad** (público o privado): público
- **Entradas requeridas**:
  - *s*: una estructura (struct) de *MATLAB*, que debe contener, como mínimo, las mismas opciones de configuración que se tomaron para crear el objeto *HomeIO* original.
- **Entradas opcionales**: (ninguna)
- **Funcionamiento esperado**: *loadobj* es un método que se ejecuta si *MATLAB* lo encuentra cuando se intenta cargar un objeto desde un fichero *.MAT*. En este caso, se prescinde de comprobar errores (que ocurrirían para archivos .MAT previos a la existencia de este método. El método simplemente extrae las configuraciones de *s* y llama al constructor de la clase. Esto es importante si se intentan guardar escuchadores, porque por sus propiedades son objetos que no se pueden copiar y causarán un *warning* al intentar restaurar este objeto desde un archivo *.MAT*. Método estático que no necesita hacer uso de ningún atributo ni método del propio objeto.
- **Salidas esperadas**: un objeto *HomeIO* de idénticas características al que fue guardado en *s*.
- **Código:**

```
function obj = loadobj(s)
            %loadobj tries to restore the previous object
            %   No need to test for errors with the isstruct check since
            %   the constructor usually generates listeners, which are
            %   non-copyable and must be restored manually (so call
            %   the constructor back with the same arguments)
            engineio_path = s.Config.engineio_path;
            data_path = s.Config.data_path;
            set_timer = s.Config.set_timer;
            set_listener = s.Config.set_listener;

            obj =
HomeIO('engineio_path',engineio_path,'data_path',data_path,'set_timer',set_timer,'set
_listener',set_listener);
        end
```

Con esto terminan las funciones contenidas en la clase de comunicación de *MATLAB* con el simulador de vivienda *Home I/O*. Más adelante en esta sección se hará una lista de las referencias que se han ido utilizando para construir esta clase, junto a algunas pruebas de funcionamiento que se lanzan para asegurar que el comportamiento de los objetos que se creen de esta clase es el de esperar.

En el Anexo B se puede encontrar la totalidad de este código, de modo que pueda ser examinado y observado en su conjunto (o se pueda copiar y pegar para obtener una versión de esta clase de comunicación). Este código está pensado para ejecutarse desde un fichero con nombre *HomeIO.m,* al tener la clase este mismo nombre.

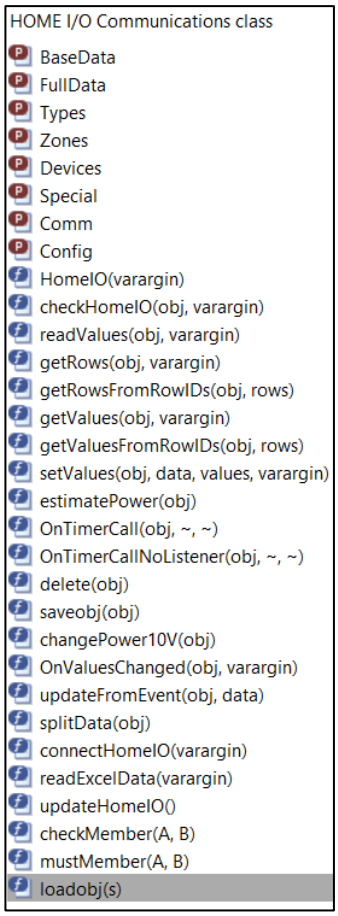


Figura 4.1 Sinóptico del contenido de la clase *HomeIO*

4.4 Ayudas y referencias utilizadas en la programación

Se ha utilizado un gran número de fuentes para ayuda y referencia en la elaboración de esta clase, que habilita de forma eficiente la comunicación entre *MATLAB* y *Home I/O*. Algunas de ellas se mencionaron anteriormente, pero se ha decidido no salpicar el código fuente ni su explicación de referencias que, en la mayoría de los casos, no aplica en un solo lugar sino que se han ido utilizando de forma continuada en el desarrollo. A continuación se muestran aquellas que han resultado de utilidad para lograr este código.

- Clases "*por valor*" y clase tipo manejador (handle): [30] [36] [31] [32] [33] [34] [37] [38]
- Uso de *struct* en clases de *MATLAB*: [35]
- Lectura de archivos *Excel* como tabla: [39]
- Trabajo con tablas: [40] [41] [42] [43] [44]
- Trabajo con temporizadores: [45] [46] [47]
- Trabajo con escuchadores de eventos (*listeners*): [48] [49] [50] [51]
- Obtención de otros procesos en ejecución desde *MATLAB*: [29] [52]
- Funciones *ismember* y *mustBeMember*: [53] [54]
- Intérprete de variables *inputparser*: [55] [56] [57] [58]
- Método *delete*: [59] [60]
- Métodos *loadobj* y *saveobj:* [61] [62] [63] [64]
- Ejemplos de Home I/O en C#: [24]

4.5 Comprobaciones de funcionamiento del código

A continuación, se lanzan algunas comprobaciones sobre funciones del código, para asegurar que la clase desarrollada es funcional y puede servir para la investigación originalmente planteada.

Para comprobar el funcionamiento del constructor de clase, se decide crear un objeto *HomeIO* con la aplicación cerrada y con la aplicación abierta. La instrucción que se lanzará será la de creación de un objeto de esta clase, tal cual: *a = HomeIO*

Se puede comprobar el *warning* que se lanza de que la aplicación no está abierta:

```
>> a = HomeIO
Warning: Unable to find Home I/O process or Home I/O running more than once. Results might be
inaccurate.
> In HomeIO/checkHomeIO (line 242)
In HomeIO (line 179)

a =

  HomeIO with properties:

    BaseData: [338×9 table]
    FullData: [338×10 table]
       Types: [1×1 struct]
       Zones: [1×1 struct]
     Devices: [1×1 struct]
     Special: [1×1 struct]
        Comm: [1×1 struct]
      Config: [1×1 struct]
```

Figura 4.2 Aviso del constructor de la clase *HomeIO* si se ejecuta con el simulador apagado

Se puede comprobar cómo el atributo *FullData* no se ha actualizado con los valores de *Home I/O*, porque ese trozo de código era de ejecución condicional a que el programa estuviera ejecutándose:

	1 RowID	2 VarType	3 MemoryType	4 DataType	5 Address	6 Zone	7 Name	8 ContactType	9 Power	10 Value
1	1	1	"Input"	"Bool"	0	"A"	"Light Swit...	"NO"	0	NaN
2	2	1	"Input"	"Bool"	1	"A"	"Light Swit...	"NO"	0	NaN
3	3	1	"Input"	"Bool"	2	"A"	"Light Swit...	"NO"	0	NaN
4	4	1	"Input"	"Bool"	3	"A"	"Up/Down ...	"NO"	0	NaN
5	5	1	"Input"	"Bool"	4	"A"	"Up/Down ...	"NO"	0	NaN
6	6	1	"Input"	"Bool"	5	"A"	"Up/Down ...	"NO"	0	NaN
7	7	1	"Input"	"Bool"	6	"A"	"Up/Down ...	"NO"	0	NaN
8	8	1	"Input"	"Bool"	7	"A"	"Light Swit...	"NO"	0	NaN
9	9	1	"Input"	"Bool"	8	"A"	"Light Swit...	"NO"	0	NaN
10	10	1	"Input"	"Bool"	9	"A"	"Light Swit...	"NO"	0	NaN

Figura 4.3 Primeras diez filas de *FullData*, sin valores, porque no se han podido actualizar

En cambio, ejecutando con el simulador abierto sí que se obtienen valores, por ejemplo la hora de simulación y el valor de los sensores de luminosidad:

```
  HomeIO with properties:

    BaseData: [338×9 table]
    FullData: [338×10 table]
       Types: [1×1 struct]
       Zones: [1×1 struct]
     Devices: [1×1 struct]
     Special: [1×1 struct]
        Comm: [1×1 struct]
      Config: [1×1 struct]

>> a.updateHomeIO
>> datetime( a.getValues(a.Devices.DateAndTimeMemory),'ConvertFrom','datenum')

ans =

  datetime

   27-Oct-2021 07:57:36

>> a.getRows(a.Devices.BrightnessSensorsFloat)

ans =

  10×10 table

    RowID    VarType    MemoryType    DataType    Address    Zone                Name                ContactType    Power    Value
    _____    _______    __________    ________    _______    ____    ____________________________    ___________    _____    _____

      148          2    "Input"       "Float"           0    "A"     "Brightness Sensor (Analogue)"    "-"            0        1.4947
      155          2    "Input"       "Float"          12    "D"     "Brightness Sensor (Analogue)"    "-"            0        0.9942
      159          2    "Input"       "Float"          24    "E"     "Brightness Sensor (Analogue)"    "-"            0        0.99479
      163          2    "Input"       "Float"          36    "F"     "Brightness Sensor (Analogue)"    "-"            0        1.4904
      168          2    "Input"       "Float"          57    "H"     "Brightness Sensor (Analogue)"    "-"            0        1.4947
      174          2    "Input"       "Float"          80    "J"     "Brightness Sensor (Analogue)"    "-"            0        1.4947
      180          2    "Input"       "Float"         103    "L"     "Brightness Sensor (Analogue)"    "-"            0        1.4947
      184          2    "Input"       "Float"         115    "M"     "Brightness Sensor (Analogue)"    "-"            0        1.4947
      188          2    "Input"       "Float"         127    "N"     "Brightness Sensor (Analogue)"    "-"            0        0.99521
      192          2    "Input"       "Float"         139    "O"     "Brightness Sensor (Analogue)"    "-"            0        5.9598
```

Figura 4.4 Muestra de la hora de la simulación y los valores de todos los sensores de luminosidad

Una vez comprobada que la lectura de datos funciona bien, se procede a realizar una escritura de datos, por ejemplo en los calefactores de la vivienda con valores aleatorios entre 0 y 10:

```
>> a.setValues(a.Devices.HeatersFloat, randi([0 10],[height(a.Devices.HeatersFloat) 1]))
>> a.updateHomeIO
>> a.getRows(a.Devices.HeatersFloat)

ans =

  12×10 table

    RowID    VarType    MemoryType    DataType    Address    Zone                Name                ContactType    Power    Value
    _____    _______    __________    ________    _______    ____    ____________________________    ___________    _____    _____

      266          5    "Output"      "Float"           1    "A"     "Heater (Analogue)"               "-"            200      5
      272          5    "Output"      "Float"          34    "D"     "Heater (Analogue)"               "-"            150      6
      274          5    "Output"      "Float"          45    "E"     "Heater (Analogue)"               "-"            175      2
      278          5    "Output"      "Float"          67    "G"     "Heater 1 (Analogue)"             "-"             75      5
      279          5    "Output"      "Float"          68    "G"     "Heater 2 (Analogue)"             "-"             75      10
      281          5    "Output"      "Float"          79    "H"     "Heater (Analogue)"               "-"            100      6
      284          5    "Output"      "Float"          91    "I"     "Heater (Analogue)"               "-"             75      5
      286          5    "Output"      "Float"         102    "J"     "Heater (Analogue)"               "-"             75      2
      288          5    "Output"      "Float"         113    "K"     "Heater (Analogue)"               "-"            100      5
      290          5    "Output"      "Float"         124    "L"     "Heater (Analogue)"               "-"             75      6
      292          5    "Output"      "Float"         135    "M"     "Heater (Analogue)"               "-"             50      7
      296          5    "Output"      "Float"         148    "N"     "Heater (Analogue)"               "-"            150      4

>> a.estimatePower

ans =

        6400
```

Figura 4.5 Escritura de los calefactores con valores enteros aleatorios y potencia estimada

La potencia estimada, como puede verse, coincide con la que ofrece la vivienda simulada:

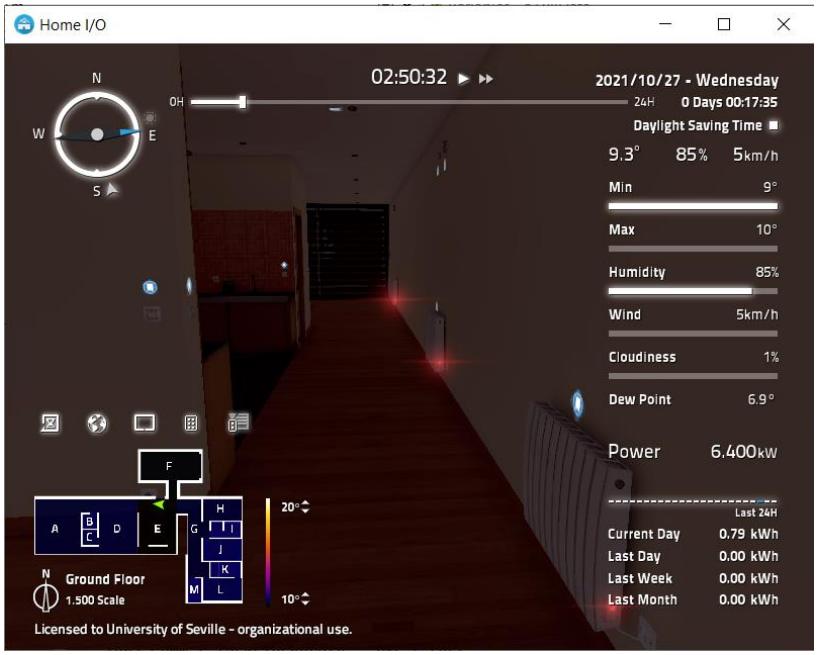


Figura 4.6 Las potencias consumidas estimada y real coinciden.

4.5 Ejercicio de aplicación: estimación de un modelo de temperatura en espacio de estados de la vivienda completa

Se quiere obtener el comportamiento térmico de la vivienda simulada completa. Para ello, se ejecuta una simulación de la vivienda, a velocidad 5000x (1 segundo real es aproximadamente 1 hora y 20 minutos de la vivienda simulada), y se encenderán los calefactores de la vivienda de forma aleatoria durante un tiempo especificado (por ejemplo, 1 día). Los datos se guardarán para usarlos posteriormente a través del *System Identification Toolbox* [65] de *MATLAB*. Dado que esto trata de ser un ejemplo sencillo, el conjunto de entrenamiento y el de verificación vendrán de la misma simulación en esta ocasión.

Se recuerda que los modelos en espacio de estados siguen estas ecuaciones:

$$\dot{x} = Ax + Bu$$

$$y = Cx + Du$$

De donde el tamaño de las matrices del experimento es sencillo de determinar a través de este simple análisis:

- **Entradas**: Las acciones sobre los calefactores de la vivienda. Hay 12 calefactores repartidos en las estancias *A, D, E, G* (2 calefactores), *H, I, J ;K, L, M, N*. Por tanto, son 12 entradas.
- **Salidas**: Las temperaturas de cada estancia. Hay 14 estancias, empezando a contar las letras desde la A (salón) a la N (oficina de primera planta), por lo que resulta en 14 salidas
- **Perturbaciones**: Una, debida a la temperatura exterior, que se ha intentado reducir a un rango mínimo de entre 9 y 10 ºC.

Los demás parámetros de simulación son como los que aparecen en la figura 4.6: temperaturas exteriores limitadas al rango indicado, humedad del 85 %, viento de 5 km/h y nubosidad de un 1 %. La simulación se ejecuta un 27 de octubre de 2021 en la latitud 37.4114 y la longitud -6.006, coordenadas en las que se sitúa la Escuela Técnica Superior de Ingenieros de Sevilla.

El código de experimentación para obtener datos de la vivienda es como el que sigue:

```
% Extract data from Home I/O

%% Simulation parameters
% Time parameters
fps = 60;
end_time = 1*24*60*60; % 1 day
get_one_frame_per_each = 1; % 1 for all frames, 2 to skip half of frames...

% Home I/O constants
m = HomeIO('set_timer',false);

home_actions = m.Devices.HeatersBool;
home_measures = [m.Devices.ZoneTemperatures; m.Devices.EnvironmentValues;
m.Devices.DateAndTimeMemory];
%home_measures = [m.Devices.ZoneTemperatures; m.Devices.EnvironmentValues];

home_data = [home_actions; home_measures];
home_data_rowIDs = home_data.RowID;

% Sample time
sample_time = get_one_frame_per_each/fps;

% How many measure frames before change (at 5000x, 1 frame is about 1.38 min)
% Current change rate: 1/10 frames ~ 15 mn
change_rate = floor(10/get_one_frame_per_each);

% Prepare variables
num_samples = end_time*sample_time;
num_inputs = height(home_actions);
num_data = height(home_data);

values = zeros(num_data,num_samples);
inputs = round(rand(num_inputs,ceil(num_samples/change_rate))); % Does not mind if
generates more values than needed

%% Simulation

tic;
for i=1:num_samples
    t1 = toc;

    % Change values if needed
    if ~mod(i,change_rate) % == 0
        % Change heaters
        actions = inputs(:,floor(i/change_rate));
        % We update, keep Home I/O open on our side and have no conflicts
        m.setValues(
home_actions,actions,'update',false,'checkHomeIO',false,'checkConflicts',false);
    end

    % Update now and reflect new values altogether
    m.updateHomeIO();

    % Get values after the interruptions
    values(:,i) = m.getValuesFromRowIDs(home_data_rowIDs);

    % Sleep time
```

```
    pause(sample_time - (toc-t1))
end

inputs_sysid = values(1:12,:)';
outputs_sysid = values(13:26,:)';

save results
```

Los resultados se introducen en la interfaz gráfica del *System Identification Toolbox* para obtener un modelo en espacio de estados dejando todos los demás parámetros por defecto. El comando de MATLAB para abrir una interfaz gráfica interactiva que permite la identificación de sistemas es *systemIdentification*, desde donde la navegación es sencilla.

Desde la interfaz del *System Identification Toolbox* quedaría algo como sigue, desde donde se ha tomado la identificación en un modelo en espacio de estados discreto y de orden 10, tal como sugiere la interfaz gráfica dejando lo demás por defecto:

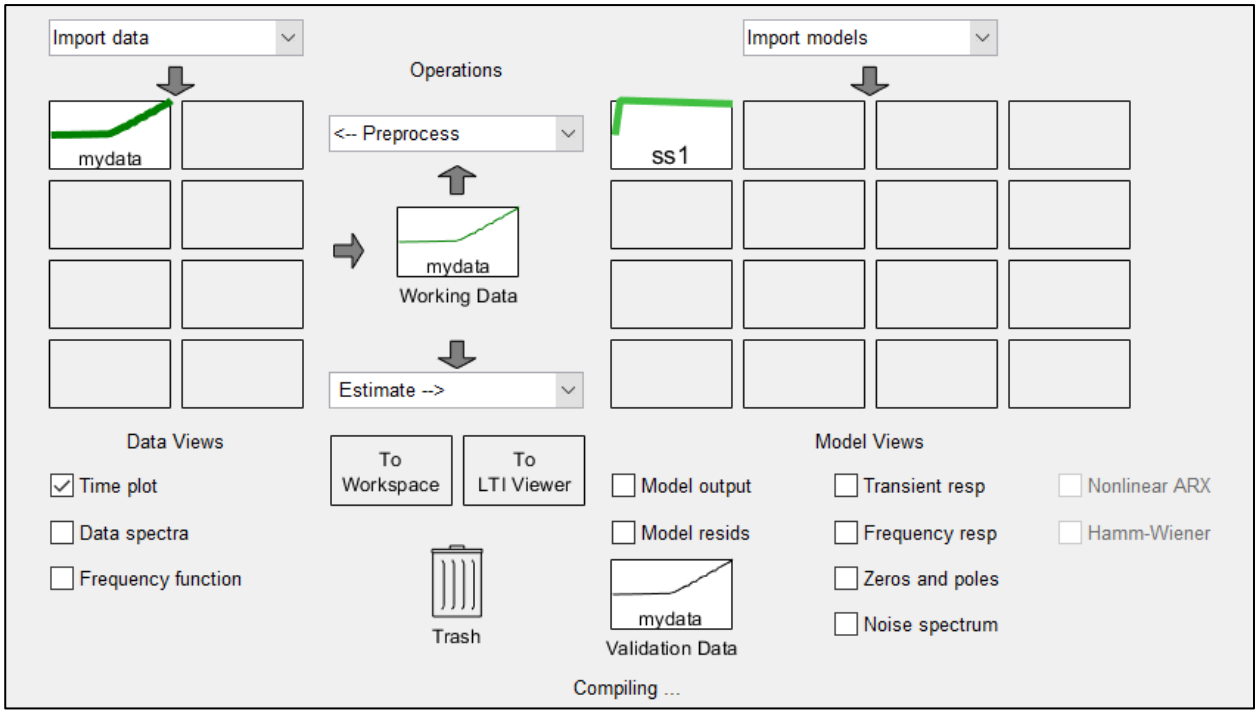


Figura 4.7 Interfaz gráfica del *System Identification Toolbox* con datos y un modelo en espacio de estados

A través de esta interfaz se puede ver de forma sencilla los datos adquiridos a través de la opción "Time plot", que aparece en la figura, donde al hacer click y marcar la casilla de verificación se puede ver el estado de salida del sistema que se pretende identificar junto con las acciones de control aplicadas a los datos y su evolución temporal. Por ejemplo, se van a mostrar los resultados únicamente para la primera variable (el salón), para evitar saturar esta memoria de gráficas (que no es el propósito de este trabajo):

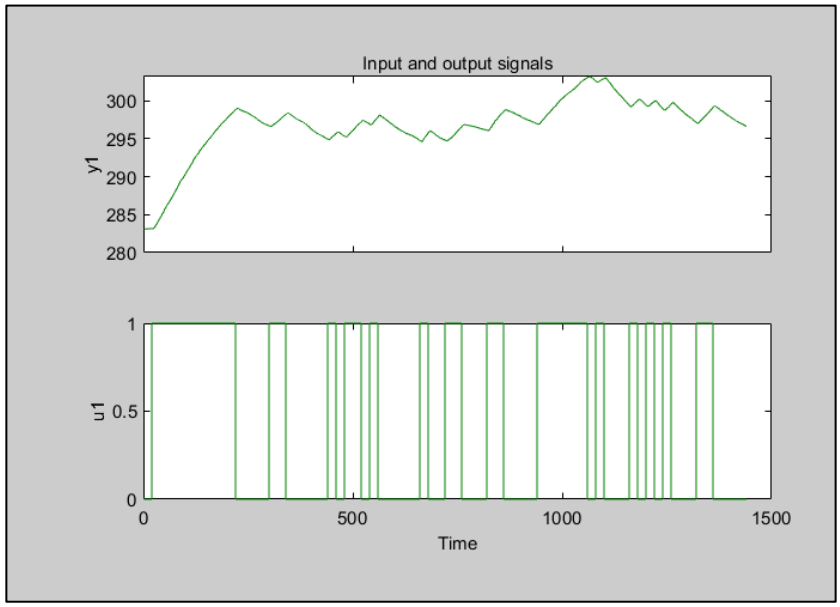


Figura 4.8 Gráfico de excitaciones y respuestas para la primera variable

La casilla de verificación "Model output", a la derecha, permite consultar el ajuste que se da a todas las variables de trabajo. Por ejemplo, también para la primera variable de salida y comparando el valor que tiene el sistema identificado frente al real obtenido a través de excitaciones sobre el sistema. Se puede ver una precisión del sistema identificado de aproximadamente un 78 %

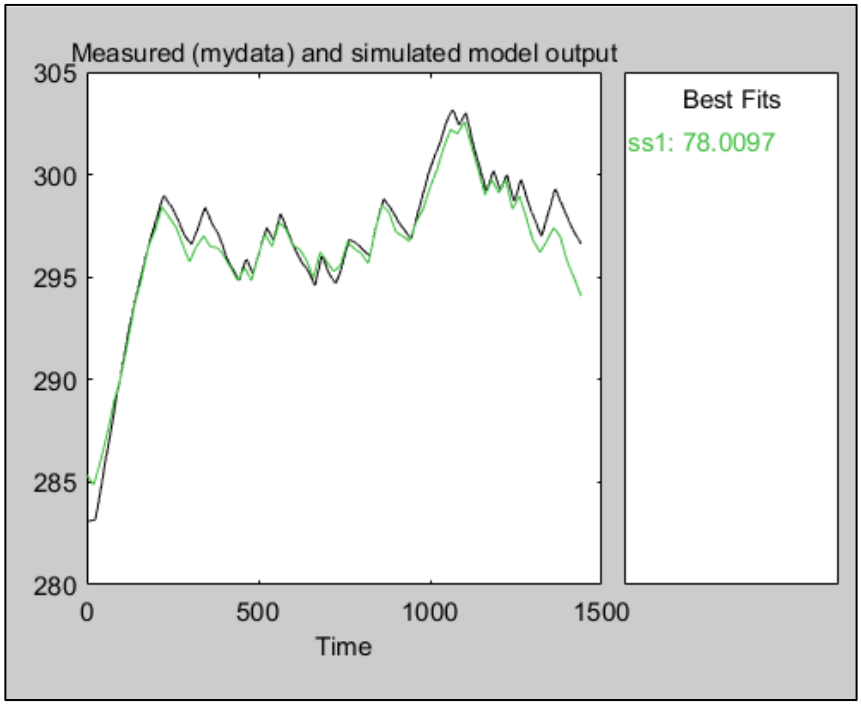


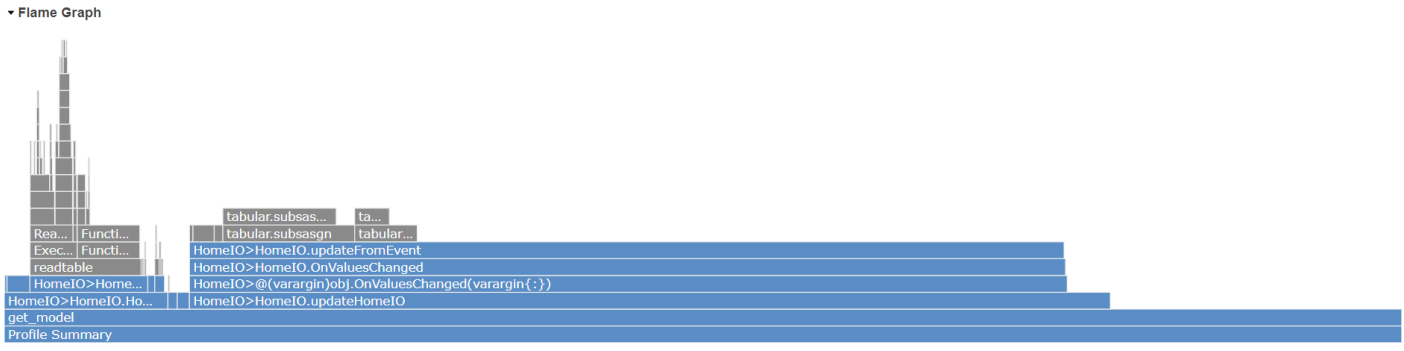
Figura 4.9 Precisión del modelo regresado en espacio de estados

Con lo que, más allá de los resultados de identificación del sistema (que han sido una excusa para la utilización de la clase desarrollada), queda demostrada la utilidad de esta plataforma diseñada para trabajar y extraer datos de *Home I/O* desde *MATLAB*.

5 DISCUSIÓN DE RESULTADOS

5.1 Problemas en los retrasos de acceso a la API de Home I/O

Uno de los problemas que más ha costado salvar es el del tiempo de ejecución: el ejemplo anterior captura los valores de cada *frame* de *Home I/O*. No obstante, es sencillo ubicar dónde se encuentran los problemas actuales y cuellos de botella del código, ejecutando el perfilador (*Profiler*) de *MATLAB*:



Generated 03-dic.-2021 11:51:38 using performance time.

Function Name	Calls	Total Time (s)	Self Time* (s)	Total Time Plot (dark band = self time)
get_model	1	34.656	7.117	
HomeIO>HomeIO.updateHomeIO	1442	22.943	1.185	
HomeIO>@(varargin)obj.OnValuesChanged(varargin{:})	2598	21.758	0.050	
HomeIO>HomeIO.OnValuesChanged	2598	21.708	0.033	
HomeIO>HomeIO.updateFromEvent	2598	21.675	16.026	

Figura 5.1 Diagrama de tiempos consumidos en la adquisición de datos de la vivienda

Se puede ver que buena parte del tiempo consumido en la ejecución se encuentra en la función *updateFromEvent*, llamada por los eventos una vez han capturado cambios en los valores de *Home I/O* al actualizar y para tomar datos de otra iteración. Examinando la función, se puede encontrar una justificación en que, aparentemente, lo que más tiempo de ejecución toma son el acceso a través de búsqueda por lógicas (*logical indexing*), pero en realidad no es así, como se puede comprobar en la figura 5.2.

El acceso lógico es relativamente rápido dentro del tiempo que tarda en ejecutarse, tal como puede comprobarse en ambas figuras, pero sí que hay un tiempo de retraso grande que parece no estar explicado por nada: son los fragmentos de barra de ejecución que no tienen "nada" encima y que *MATLAB* toma como "*Self time (built-ins, overhead, etc.)*".

La realidad es que este tiempo "propio" es el retraso que proporciona el uso de la *API* de *Home I/O*, entre retrasos de *MATLAB* al utilizar *.NET* y el tiempo que tarda esta API en responder a las peticiones que se le hace, el cual es (por desgracia) inevitable, y ocurre en cada función que interactúa con el mapa de memoria al menos una o más veces.

Lines where the most time was spent

Line Number	Code	Calls	Total Time	% Time	Time Plot
631	`obj.Types.Floats.MemoryType ==...`	48455	5.533 s	25.5%	
633	`) = datarow.Value;`	48455	5.244 s	24.2%	
632	`obj.Types.Floats.Address == da...`	48455	4.166 s	19.2%	
622	`datarow = data.MemoriesFloat(i...`	48455	3.935 s	18.2%	
645	`temp = double([temp.Year temp....`	2598	0.494 s	2.3%	
All other lines			2.304 s	10.6%	
Totals			21.675 s	100%	

Children (called functions)

Function Name	Function Type	Calls	Total Time	% Time	Time Plot
tabular.subsasgn	function	51057	3.218 s	14.8%	
tabular.subsref	function	102114	1.576 s	7.3%	
tabular.dotParenReference	function	51057	0.525 s	2.4%	
tabular.numArgumentsFromSubscript	function	102114	0.239 s	1.1%	
datenum	function	2598	0.091 s	0.4%	
Self time (built-ins, overhead, etc.)			16.026 s	73.9%	
Totals			21.675 s	100%	

Figura 5.2 Tiempo que empleado por *MATLAB* en acceder a cada línea dentro de *updateFromEvent*

Las gráficas de este tipo, en que el tiempo "propio" consume gran parte del tiempo en la ejecución de cada función son una constante, para cada método que usa el acceso a las *API* de *Home I/O*.

Esto quiere indicar que el código desarrollado es relativamente eficiente, a pesar de que acceder a cualquiera de los servicios proporcionados por *Home I/O* supone acometer siempre un retraso considerable en la ejecución, aunque afortunadamente sí que permite una comunicación adecuada entre ambos procesos.

5.2 Elección de alternativas

Como puede ser de esperar, la clase de comunicaciones entre plataformas no se ha desarrollado de la nada. Se han ensayado varias implementaciones previas, que no han pasado un filtro de calidad.

Se han ensayado numerosos métodos de acceso en todas las funciones y seleccionado aquellas que son más eficientes computacionalmente, para que el uso de esta herramienta de comunicación sea lo más rápida posible, descartando las alternativas menos eficientes. Por ejemplo, para el caso del método *updateFromEvent*, se puede ver en la siguiente figura que la opción elegida (bajo el comentario "*super newer method*") que se ha implementado, con algunas modificaciones posteriores, es la más eficiente para que el código resulte más rápido y mitigue en medida de lo posible los retrasos inducidos por otros componentes, porque otras alternativas ya fueron descartadas a estas alturas del desarrollo:

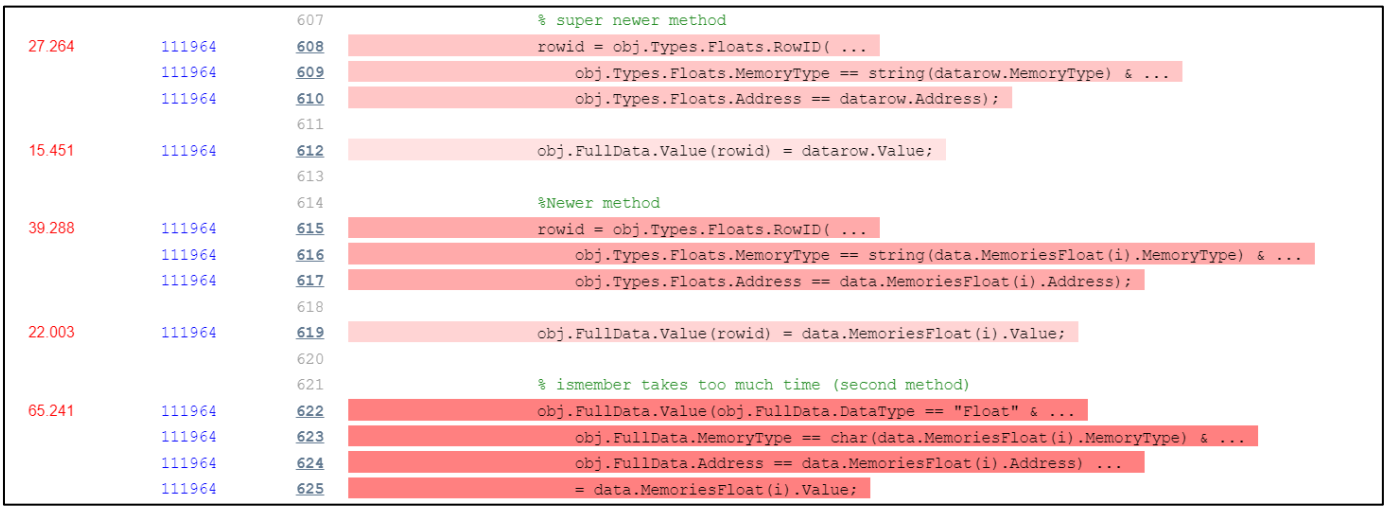


Figura 5.3 Comparación entre algunas alternativas antes de la versión final

5.2 Conclusiones y trabajos futuros

Como conclusión principal se tiene que se ha desarrollado una herramienta valiosa, que puede servir para investigación y sencilla de usar, con una buena capacidad de acceso a datos y obtención rápida del estado en que se encuentra la casa, que simplifica muchísimo otras tareas que vayan a realizarse.

Se espera que este trabajo sirva para continuar otros trabajos de investigación que se desarrollen utilizando *Home I/O* para validar alguna hipótesis previa y aprovechando que se puede acelerar la simulación a x5000 para obtención más rápida de resultados. Esa última es una característica que, si bien la realidad material no la ofrece, sí que en algunas ocasiones las personas podrían echar en falta para conseguir algún objetivo.

REFERENCIAS

[1] Cisco Systems, Inc., 2013. [En línea]. Available: https://www.cisco.com/c/dam/en_us/about/business-insights/docs/ioe-value-at-stake-public-sector-analysis-faq.pdf. [Último acceso: 28 11 2021].

[2] D. Evans, «Cisco Blog,» Cisco Systems, Inc., 7 11 2012. [En línea]. Available: https://web.archive.org/web/20190907035910/https://blogs.cisco.com/digital/how-the-internet-of-everything-will-change-the-worldfor-the-better-infographic. [Último acceso: 07 09 2019].

[3] L. Lik-Hang, T. Braud, P. Zhou, L. Wang, D. Xu, Z. Lin, A. Kumar, C. Bermejo y P. Hui, «All One Needs to Know about Metaverse: A Complete Survey on Technological Singularity, Virtual Ecosystem, and Research Agenda,» 2021.

[4] H. Touqeer, S. Zaman, R. Amin, M. Hussain, F. Al-Turjman y M. Bilal, «Smart home security: challenges, issues and solutions at different IoT layers,» *The Journal of Supercomputing,* nº 77, p. 14053–14089, 2021.

[5] S. Kumar, P. Tiwari y M. Zymbler, «Internet of Things is a revolutionary approach for future technology enhancement: a review,» *Journal of Big Data,* nº 6, p. 111, 2019.

[6] V. Orfanos, S. D. Kaminaris, D. Piromalis y P. Papageorgas, «Trends in home automation systems and protocols,» *AIP Conference Proceedings,* vol. 2190, nº 1, 2019.

[7] V. A. Orfanos, S. D. Kaminaris, D. Piromalis y P. Papageorgas, «Smart home automation in the IoT era: A communication technologies review,» *AIP Conference Proceedings ,* vol. 2307, nº 1, 2020.

[8] Operador del Mercado Ibérico, Polo Español, SA, «OMIE,» 11 28 2021. [En línea]. Available: https://www.omie.es/es. [Último acceso: 11 28 2021].

[9] J. Liboreiro y A. d. Filippis, «euronews,» 28 10 2021. [En línea]. Available: https://www.euronews.com/2021/10/28/why-europe-s-energy-prices-are-soaring-and-could-get-much-worse. [Último acceso: 29 11 2021].

[10] S. Elliott, A. Franke, F. Watson y H. Edwardes-Evans, «S&P Global,» 30 09 2021. [En línea]. Available: https://www.spglobal.com/platts/en/market-insights/latest-news/natural-gas/093021-infographic-russia-europe-gas-prices-winter-ukraine-lng-coal-power-prices-nord-stream-cop26. [Último acceso: 29 11 2021].

[11] F. Carita, «LevelTen Energy,» 21 10 2021. [En línea]. Available: https://www.leveltenenergy.com/post/europe-energy-crisis. [Último acceso: 29 11 2021].

[12] MathWorks, «MathWorks - Creadores de Matlab y Simulink,» [En línea]. Available: https://es.mathworks.com/. [Último acceso: 29 11 2021].

[13] AlternativeTo, «AlternativeTo, crowdsourced software recommendations,» [En línea]. Available: https://alternativeto.net/software/matlab/. [Último acceso: 29 11 2021].

[14] Real Games, «Home I/O - Smart Home Simulation - Real Games,» [En línea]. Available: https://realgames.co/home-io. [Último acceso: 29 11 2021].

[15] Real Games, «3D Simulation Software,» 29 11 2021. [En línea]. Available: https://realgames.co/. [Último acceso: 29 11 2021].

[16] Centre de Recherche en STIC, «Bienvenue | CRESTIC | URCA | laboratoire de recherche,» [En línea]. Available: https://crestic.univ-reims.fr/fr/. [Último acceso: 29 11 2021].

[17] Real Games, «Home I/O,» [En línea]. Available: https://docs.realgames.co/homeio/en/. [Último acceso: 29 11 2021].

[18] Real Games, «Controls - Home I/O,» [En línea]. Available: https://docs.realgames.co/homeio/en/controls/. [Último acceso: 29 11 2021].

[19] Real Games, «Thermal Behavior - Home I/O,» 29 11 2021. [En línea]. Available: https://docs.realgames.co/homeio/en/thermal-behavior/. [Último acceso: 29 11 2021].

[20] Real Games, «Connecting to External Technologies - Home I/O,» [En línea]. Available: https://docs.realgames.co/homeio/en/connectio/. [Último acceso: 29 11 2021].

[21] Real Games, «Devices Map - Home I/O,» [En línea]. Available: https://docs.realgames.co/homeio/en/devices-map/. [Último acceso: 29 11 2021].

[22] Real Games, «Device Modes,» [En línea]. Available: https://docs.realgames.co/homeio/en/device-modes/. [Último acceso: 30 11 2021].

[23] Real Games, «Connect I/O,» [En línea]. Available: https://docs.realgames.co/connectio/. [Último acceso: 30 11 2021].

[24] Real Games, «Getting Started - Home I/O,» [En línea]. Available: https://docs.realgames.co/homeio/en/sdk-getting-started/. [Último acceso: 30 11 2021].

[25] Real Games, «Memory Addresses - Home I/O,» [En línea]. Available: https://docs.realgames.co/homeio/en/memory-addresses/. [Último acceso: 30 11 2021].

[26] Real Games, «Home I/O utiliser MATLAB/Simulink pour réguler l'intensité lumineuse d'une pièce (Software in the loop) - Home I/O,» [En línea]. Available: https://docs.realgames.co/homeio/fr/using-matlab/. [Último acceso: 30 11 2021].

[27] E. Bravo del Cid, «Gestión activa de la demanda eléctrica mediante sistema domótico. (Trabajo Fin de Grado Inédito),» Sevilla, Universidad de Sevilla, 2021.

[28] J. D. Cook, «Cosmic rays flipping bits,» 29 05 2019. [En línea]. Available: https://www.johndcook.com/blog/2019/05/20/cosmic-rays-flipping-bits/. [Último acceso: 01 12 2021].

[29] Stack Overflow, «Detect matlab processes from within matlab,» 13 02 2009. [En línea]. Available: https://stackoverflow.com/questions/858301/detect-matlab-processes-from-within-matlab. [Último acceso: 02 12 2021].

[30] MathWorks, «Class definition keywords,» [En línea]. Available:

https://www.mathworks.com/help/matlab/ref/classdef.html?s_tid=doc_ta. [Último acceso: 02 12 2021].

[31] MathWorks, «Superclass of all handle classes,» [En línea]. Available: https://www.mathworks.com/help/matlab/ref/handle-class.html. [Último acceso: 02 12 2021].

[32] MathWorks, «Comparison of Handle and Value Classes,» [En línea]. Available: https://www.mathworks.com/help/matlab/matlab_oop/comparing-handle-and-value-classes.html. [Último acceso: 02 12 2021].

[33] MathWorks, «Handle Object Behavior,» [En línea]. Available: https://www.mathworks.com/help/matlab/matlab_oop/handle-objects.html. [Último acceso: 02 12 2021].

[34] MathWorks, «Events and Listeners Syntax,» [En línea]. Available: https://www.mathworks.com/help/matlab/matlab_oop/events-and-listeners-syntax-and-techniques.html. [Último acceso: 02 12 2021].

[35] MATLAB Answers, «How to use structures as properties of a matlab class ?,» [En línea]. Available: https://www.mathworks.com/matlabcentral/answers/223848-how-to-use-structures-as-properties-of-a-matlab-class. [Último acceso: 02 12 2021].

[36] MATLAB Answers, «Updating property of an Object without creating new object,» 15 03 2015. [En línea]. Available: https://www.mathworks.com/matlabcentral/answers/183246-updating-property-of-an-object-without-creating-new-object. [Último acceso: 02 12 2021].

[37] MathWorks, «Class Components,» [En línea]. Available: https://www.mathworks.com/help/matlab/matlab_oop/class-components.html. [Último acceso: 02 11 2021].

[38] MathWorks, «Creating a Simple Class,» [En línea]. Available: https://www.mathworks.com/help/matlab/matlab_oop/create-a-simple-class.html. [Último acceso: 02 11 2021].

[39] MathWorks, «Create table from file,» [En línea]. Available: https://www.mathworks.com/help/matlab/ref/readtable.html. [Último acceso: 02 12 2021].

[40] MathWorks, «Table array with named variables that can contain different types,» [En línea]. Available: https://www.mathworks.com/help/matlab/ref/table.html. [Último acceso: 02 11 2021].

[41] MathWorks, «Advantages of Using Tables,» [En línea]. Available: https://www.mathworks.com/help/matlab/matlab_prog/advantages-of-using-tables.html. [Último acceso: 02 11 2021].

[42] MathWorks, «Add, Delete, and Rearrange Table Variables,» [En línea]. Available: https://www.mathworks.com/help/matlab/matlab_prog/add-and-delete-table-variables.html. [Último acceso: 02 11 2021].

[43] MathWorks, «Access Data in Tables,» [En línea]. Available: https://www.mathworks.com/help/matlab/matlab_prog/access-data-in-a-table.html. [Último acceso: 02 11 2021].

[44] MathWorks, «Find Array Elements That Meet a Condition,» [En línea]. Available: https://www.mathworks.com/help/matlab/matlab_prog/find-array-elements-that-meet-a-condition.html.

[Último acceso: 02 11 2021].

[45] MathWorks, «Schedule execution of MATLAB commands,» [En línea]. Available: https://www.mathworks.com/help/matlab/ref/timer.html. [Último acceso: 02 12 2021].

[46] MathWorks, «Timer Callback Functions,» [En línea]. Available: https://www.mathworks.com/help/matlab/matlab_prog/timer-callback-functions.html. [Último acceso: 02 11 2021].

[47] MathWorks, «Handling Timer Queuing Conflicts,» [En línea]. Available: https://www.mathworks.com/help/matlab/matlab_prog/handling-timer-queuing-conflicts.html. [Último acceso: 02 11 2021].

[48] MathWorks, «Create event listener without binding to event source,» [En línea]. Available: https://www.mathworks.com/help/matlab/ref/handle.listener.html. [Último acceso: 02 12 2021].

[49] MathWorks, «Overview Events and Listeners,» [En línea]. Available: https://www.mathworks.com/help/matlab/matlab_oop/learning-to-use-events-and-listeners.html. [Último acceso: 02 12 2021].

[50] Stack Overflow, «Why does my listener get deleted when I saveobj()?,» 10 04 2014. [En línea]. Available: https://stackoverflow.com/questions/22998367/why-does-my-listener-get-deleted-when-i-saveobj. [Último acceso: 02 12 2021].

[51] MathWorks, «Restore Listeners,» [En línea]. Available: https://www.mathworks.com/help/matlab/matlab_oop/save-and-load-objects-with-listeners.html. [Último acceso: 02 12 2021].

[52] MATLAB Answers, «How to find the Process ID (PID) in matlab,» 08 06 2012. [En línea]. Available: https://www.mathworks.com/matlabcentral/answers/40617-how-to-find-the-process-id-pid-in-matlab#answer_238529. [Último acceso: 02 12 2021].

[53] MathWorks, «Array elements that are members of set array,» [En línea]. Available: https://www.mathworks.com/help/matlab/ref/double.ismember.html. [Último acceso: 02 12 2021].

[54] MathWorks, «Validate that value is member of specified set,» [En línea]. Available: https://www.mathworks.com/help/matlab/ref/mustbemember.html. [Último acceso: 02 12 2021].

[55] MathWorks, «Input parser for Functions,» [En línea]. Available: https://www.mathworks.com/help/matlab/ref/inputparser.html. [Último acceso: 02 12 2021].

[56] MathWorks, «Add optional name-value pair argument into input parser scheme,» [En línea]. Available: https://www.mathworks.com/help/matlab/ref/inputparser.addparameter.html. [Último acceso: 02 12 2021].

[57] MathWorks, «Add required, positional argument into input parser scheme,» [En línea]. Available: https://www.mathworks.com/help/matlab/ref/inputparser.addrequired.html. [Último acceso: 02 12 2021].

[58] MathWorks, «Parse function inputs,» [En línea]. Available: https://www.mathworks.com/help/matlab/ref/inputparser.parse.html. [Último acceso: 02 12 2021].

[59] MathWorks, «Handle Class Destructor,» [En línea]. Available:

https://www.mathworks.com/help/matlab/matlab_oop/handle-class-destructors.html. [Último acceso: 02 12 2021].

[60] MathWorks, «Delete handle object,» [En línea]. Available: https://www.mathworks.com/help/matlab/ref/handle.delete.html. [Último acceso: 02 12 2021].

[61] MathWorks, «Optimize the object save and load process,» [En línea]. Available: https://www.mathworks.com/help/matlab/save-and-load.html. [Último acceso: 02 12 2021].

[62] MathWorks, «Modify save process for object,» [En línea]. Available: https://www.mathworks.com/help/matlab/ref/saveobj.html. [Último acceso: 02 12 2021].

[63] MathWorks, «Customize load process for objects,» [En línea]. Available: https://www.mathworks.com/help/matlab/ref/loadobj.html. [Último acceso: 02 12 2021].

[64] MathWorks, «Basic saveobj and loadobj Pattern,» [En línea]. Available: https://www.mathworks.com/help/matlab/matlab_oop/flexible-saveobj-and-loadobj-pattern.html. [Último acceso: 02 12 2021].

[65] MathWorks, «Identify Linear Models Using System Identification App,» [En línea]. Available: https://www.mathworks.com/help/ident/gs/identify-linear-models-using-the-gui.html. [Último acceso: 02 12 2021].

ANEXO A: LISTA COMPLETA DE DISPOSITIVOS DE HOME I/O

Memory Type	Data Type	Address	Zone	Name	Contact Type	Power
Input	Bool	0	A	Light Switch 1	NO	0
Input	Bool	1	A	Light Switch 2	NO	0
Input	Bool	2	A	Light Switch 3	NO	0
Input	Bool	3	A	Up/Down Switch 1 (Up)	NO	0
Input	Bool	4	A	Up/Down Switch 1 (Down)	NO	0
Input	Bool	5	A	Up/Down Switch 2 (Up)	NO	0
Input	Bool	6	A	Up/Down Switch 2 (Down)	NO	0
Input	Bool	7	A	Light Switch Dimmer 1 (Up)	NO	0
Input	Bool	8	A	Light Switch Dimmer 1 (Down)	NO	0
Input	Bool	9	A	Light Switch Dimmer 2 (Up)	NO	0
Input	Bool	10	A	Light Switch Dimmer 2 (Down)	NO	0
Input	Bool	11	A	Light Switch Dimmer 3 (Up)	NO	0
Input	Bool	12	A	Light Switch Dimmer 3 (Down)	NO	0
Input	Bool	13	A	Door Detector 1	NO	0
Input	Bool	14	A	Door Detector 2	NO	0
Input	Bool	15	A	Motion Detector	NO	0
Input	Bool	16	A	Brightness Sensor	NC	0
Input	Bool	17	A	Smoke Detector	NO	0
Input	Bool	27	B	Light Switch 1	NO	0

Memory Type	Data Type	Address	Zone	Name	Contact Type	Power
Input	Bool	28	B	Light Switch 2	NO	0
Input	Bool	29	B	Motion Detector	NO	0
Input	Bool	39	C	Light Switch	NO	0
Input	Bool	49	D	Up/Down Switch (Up)	NO	0
Input	Bool	50	D	Up/Down Switch (Down)	NO	0
Input	Bool	51	D	Light Switch Dimmer 1 (Up)	NO	0
Input	Bool	52	D	Light Switch Dimmer 1 (Down)	NO	0
Input	Bool	53	D	Light Switch Dimmer 2 (Up)	NO	0
Input	Bool	54	D	Light Switch Dimmer 2 (Down)	NO	0
Input	Bool	55	D	Door Detector	NO	0
Input	Bool	56	D	Motion Detector	NO	0
Input	Bool	57	D	Brightness Sensor	NC	0
Input	Bool	67	E	Light Switch 1	NO	0
Input	Bool	68	E	Light Switch 2	NO	0
Input	Bool	69	E	Light Switch 3	NO	0
Input	Bool	70	E	Light Switch 4	NO	0
Input	Bool	71	E	Light Switch 5	NO	0
Input	Bool	72	E	Up/Down Switch (Up)	NO	0
Input	Bool	73	E	Up/Down Switch (Down)	NO	0
Input	Bool	74	E	Light Switch Dimmer 1 (Up)	NO	0
Input	Bool	75	E	Light Switch Dimmer 1 (Down)	NO	0
Input	Bool	76	E	Light Switch Dimmer 2 (Up)	NO	0

Memory Type	Data Type	Address	Zone	Name	Contact Type	Power
Input	Bool	77	E	Light Switch Dimmer 2 (Down)	NO	0
Input	Bool	78	E	Door Detector 1	NO	0
Input	Bool	79	E	Door Detector 2	NO	0
Input	Bool	80	E	Motion Detector	NO	0
Input	Bool	81	E	Brightness Sensor	NC	0
Input	Bool	82	E	Alarm Key Pad (Armed)	NO	0
Input	Bool	92	F	Light Switch 1	NO	0
Input	Bool	93	F	Light Switch 2	NO	0
Input	Bool	94	F	Up/Down Switch 1 (Up)	NO	0
Input	Bool	95	F	Up/Down Switch 1 (Down)	NO	0
Input	Bool	96	F	Up/Down Switch 2 (Up)	NO	0
Input	Bool	97	F	Up/Down Switch 2 (Down)	NO	0
Input	Bool	98	F	Motion Detector	NO	0
Input	Bool	99	F	Brightness Sensor	NO	0
Input	Bool	100	F	Garage Door (Opened)	NO	0
Input	Bool	101	F	Garage Door (Closed)	NO	0
Input	Bool	102	F	Garage Door (Infrared)	NO	0
Input	Bool	112	G	Light Switch 1	NO	0
Input	Bool	113	G	Light Switch 2	NO	0
Input	Bool	114	G	Light Switch 3	NO	0
Input	Bool	115	G	Light Switch 4	NO	0
Input	Bool	116	G	Light Switch 5	NO	0
Input	Bool	117	G	Up/Down Switch (Up)	NO	0

Memory Type	Data Type	Address	Zone	Name	Contact Type	Power
Input	Bool	118	G	Up/Down Switch (Down)	NO	0
Input	Bool	119	G	Motion Detector	NO	0
Input	Bool	120	G	Smoke Detector	NO	0
Input	Bool	130	H	Light Switch 1	NO	0
Input	Bool	131	H	Light Switch 2	NO	0
Input	Bool	132	H	Up/Down Switch (Up)	NO	0
Input	Bool	133	H	Up/Down Switch (Down)	NO	0
Input	Bool	134	H	Light Switch Dimmer (Up)	NO	0
Input	Bool	135	H	Light Switch Dimmer (Down)	NO	0
Input	Bool	136	H	Door Detector 1	NO	0
Input	Bool	137	H	Door Detector 2	NO	0
Input	Bool	138	H	Motion Detector	NO	0
Input	Bool	139	H	Brightness Sensor	NC	0
Input	Bool	140	H	Smoke Detector	NO	0
Input	Bool	150	I	Light Switch	NO	0
Input	Bool	151	I	Light Switch Dimmer (Up)	NO	0
Input	Bool	152	I	Light Switch Dimmer (Down)	NO	0
Input	Bool	153	I	Motion Detector	NO	0
Input	Bool	163	J	Light Switch 1	NO	0
Input	Bool	164	J	Light Switch 2	NO	0
Input	Bool	165	J	Up/Down Switch (Up)	NO	0
Input	Bool	166	J	Up/Down Switch (Down)	NO	0

Memory Type	Data Type	Address	Zone	Name	Contact Type	Power
Input	Bool	167	J	Light Switch Dimmer (Up)	NO	0
Input	Bool	168	J	Light Switch Dimmer (Down)	NO	0
Input	Bool	169	J	Door Detector 1	NO	0
Input	Bool	170	J	Door Detector 2	NO	0
Input	Bool	171	J	Motion Detector	NO	0
Input	Bool	172	J	Brightness Sensor	NC	0
Input	Bool	173	J	Smoke Detector	NO	0
Input	Bool	183	K	Light Switch Dimmer (Up)	NO	0
Input	Bool	184	K	Light Switch Dimmer (Down)	NO	0
Input	Bool	185	K	Motion Detector	NO	0
Input	Bool	195	L	Light Switch 1	NO	0
Input	Bool	196	L	Light Switch 2	NO	0
Input	Bool	197	L	Up/Down Switch (Up)	NO	0
Input	Bool	198	L	Up/Down Switch (Down)	NO	0
Input	Bool	199	L	Light Switch Dimmer 1 (Up)	NO	0
Input	Bool	200	L	Light Switch Dimmer 1 (Down)	NO	0
Input	Bool	201	L	Light Switch Dimmer 2 (Up)	NO	0
Input	Bool	202	L	Light Switch Dimmer 2 (Down)	NO	0
Input	Bool	203	L	Door Detector 1	NO	0
Input	Bool	204	L	Door Detector 2	NO	0
Input	Bool	205	L	Motion Detector	NO	0
Input	Bool	206	L	Brightness Sensor	NC	0
Input	Bool	207	L	Smoke Detector	NO	0

Memory Type	Data Type	Address	Zone	Name	Contact Type	Power
Input	Bool	217	M	Light Switch	NO	0
Input	Bool	218	M	Up/Down Switch (Up)	NO	0
Input	Bool	219	M	Up/Down Switch (Down)	NO	0
Input	Bool	220	M	Door Detector 1	NO	0
Input	Bool	221	M	Door Detector 2	NO	0
Input	Bool	222	M	Motion Detector	NO	0
Input	Bool	223	M	Brightness Sensor	NC	0
Input	Bool	233	N	Light Switch 1	NO	0
Input	Bool	234	N	Light Switch 2	NO	0
Input	Bool	235	N	Light Switch 3	NO	0
Input	Bool	236	N	Light Switch Dimmer 1 (Up)	NO	0
Input	Bool	237	N	Light Switch Dimmer 1 (Down)	NO	0
Input	Bool	238	N	Light Switch Dimmer 2 (Up)	NO	0
Input	Bool	239	N	Light Switch Dimmer 2 (Down)	NO	0
Input	Bool	240	N	Light Switch Dimmer 3 (Up)	NO	0
Input	Bool	241	N	Light Switch Dimmer 3 (Down)	NO	0
Input	Bool	242	N	Up/Down Switch (Up)	NO	0
Input	Bool	243	N	Up/Down Switch (Down)	NO	0
Input	Bool	244	N	Door Detector 1	NO	0
Input	Bool	245	N	Door Detector 2	NO	0
Input	Bool	246	N	Motion Detector	NO	0
Input	Bool	247	N	Brightness Sensor	NC	0

Memory Type	Data Type	Address	Zone	Name	Contact Type	Power
Input	Bool	248	N	Smoke Detector	NO	0
Input	Bool	258	O	Motion Detector	NO	0
Input	Bool	259	O	Brightness Sensor	NC	0
Input	Bool	260	O	Entrance Gate (Opened)	NO	0
Input	Bool	261	O	Entrance Gate (Closed)	NO	0
Input	Bool	262	O	Entrance Gate (Infrared 1)	NO	0
Input	Bool	263	O	Entrance Gate (Infrared 2)	NO	0
Input	Bool	264	O	Entrance Gate (Infrared 3)	NO	0
Input	Bool	274	-	Remote Button 1	NO	0
Input	Bool	275	-	Remote Button 2	NO	0
Input	Bool	276	-	Remote Button 3	NO	0
Input	Bool	277	-	Remote Button 4	NO	0
Input	Bool	278	-	Remote Button 5	NO	0
Input	Bool	279	-	Remote Button 6	NO	0
Input	Bool	280	-	Remote Button 7	NO	0
Input	Bool	281	-	Remote Button 8	NO	0
Input	Float	0	A	Brightness Sensor (Analogue)	-	0
Input	Float	1	A	Thermostat (Room Temperature)	-	0
Input	Float	2	A	Thermostat (Set Point)	-	0
Input	Float	3	A	Roller Shades (Openness)	-	0
Input	Float	4	A	Roller Shades (Openness)	-	0
Input	Float	5	A	Roller Shades (Openness)	-	0
Input	Float	6	A	Roller Shades (Openness)	-	0

Memory Type	Data Type	Address	Zone	Name	Contact Type	Power
Input	Float	12	D	Brightness Sensor (Analogue)	-	0
Input	Float	13	D	Thermostat (Room Temperature)	-	0
Input	Float	14	D	Thermostat (Set Point)	-	0
Input	Float	15	D	Roller Shades (Openness)	-	0
Input	Float	24	E	Brightness Sensor (Analogue)	-	0
Input	Float	25	E	Thermostat (Room Temperature)	-	0
Input	Float	26	E	Thermostat (Set Point)	-	0
Input	Float	27	E	Roller Shades (Openness)	-	0
Input	Float	36	F	Brightness Sensor (Analogue)	-	0
Input	Float	37	F	Roller Shades (Openness)	-	0
Input	Float	46	G	Thermostat (Room Temperature)	-	0
Input	Float	47	G	Thermostat (Set Point)	-	0
Input	Float	48	G	Roller Shades (Openness)	-	0
Input	Float	57	H	Brightness Sensor (Analogue)	-	0
Input	Float	58	H	Thermostat (Room Temperature)	-	0
Input	Float	59	H	Thermostat (Set Point)	-	0
Input	Float	60	H	Roller Shades (Openness)	-	0
Input	Float	69	I	Thermostat (Room Temperature)	-	0
Input	Float	70	I	Thermostat (Set Point)	-	0
Input	Float	80	J	Brightness Sensor (Analogue)	-	0
Input	Float	81	J	Thermostat (Room Temperature)	-	0
Input	Float	82	J	Thermostat (Set Point)	-	0

Memory Type	Data Type	Address	Zone	Name	Contact Type	Power
Input	Float	83	J	Roller Shades (Openness)	-	0
Input	Float	92	K	Thermostat (Room Temperature)	-	0
Input	Float	93	K	Thermostat (Set Point)	-	0
Input	Float	103	L	Brightness Sensor (Analogue)	-	0
Input	Float	104	L	Thermostat (Room Temperature)	-	0
Input	Float	105	L	Thermostat (Set Point)	-	0
Input	Float	106	L	Roller Shades (Openness)	-	0
Input	Float	115	M	Brightness Sensor (Analogue)	-	0
Input	Float	116	M	Thermostat (Room Temperature)	-	0
Input	Float	117	M	Thermostat (Set Point)	-	0
Input	Float	118	M	Roller Shades (Openness)	-	0
Input	Float	127	N	Brightness Sensor (Analogue)	-	0
Input	Float	128	N	Thermostat (Room Temperature)	-	0
Input	Float	129	N	Thermostat (Set Point)	-	0
Input	Float	130	N	Roller Shades (Openness)	-	0
Input	Float	139	O	Brightness Sensor (Analogue)	-	0
Input	DateTime	0	-	Date and Time	-	0
Output	Bool	0	A	Lights	-	60
Output	Float	0	A	Lights (Analogue)	-	60
Output	Bool	1	A	Roller Shades 1 (Up)	-	10
Output	Float	1	A	Heater (Analogue)	-	2000
Output	Bool	2	A	Roller Shades 1 (Down)	-	10
Output	Bool	3	A	Roller Shades 2 (Up)	-	10

Memory Type	Data Type	Address	Zone	Name	Contact Type	Power
Output	Bool	4	A	Roller Shades 2 (Down)	-	10
Output	Bool	5	A	Roller Shades 3 (Up)	-	10
Output	Bool	6	A	Roller Shades 3 (Down)	-	10
Output	Bool	7	A	Roller Shades 4 (Up)	-	10
Output	Bool	8	A	Roller Shades 4 (Down)	-	10
Output	Bool	9	A	Heater	-	2000
Output	Float	11	B	Lights 1 (Analogue)	-	10
Output	Float	12	B	Lights 2 (Analogue)	-	10
Output	Bool	19	B	Lights 1	-	10
Output	Bool	20	B	Lights 2	-	10
Output	Float	22	C	Lights (Analogue)	-	10
Output	Bool	30	C	Lights	-	10
Output	Float	32	D	Lights 1 (Analogue)	-	60
Output	Float	33	D	Lights 2 (Analogue)	-	30
Output	Float	34	D	Heater (Analogue)	-	1500
Output	Bool	40	D	Lights 1	-	60
Output	Bool	41	D	Lights 2	-	30
Output	Bool	42	D	Roller Shades (Up)	-	10
Output	Bool	43	D	Roller Shades (Down)	-	10
Output	Bool	44	D	Heater	-	1500
Output	Float	44	E	Lights (Analogue)	-	60
Output	Float	45	E	Heater (Analogue)	-	1750

Memory Type	Data Type	Address	Zone	Name	Contact Type	Power
Output	Bool	54	E	Lights	-	60
Output	Bool	55	E	Roller Shades (Up)	-	10
Output	Float	55	F	Lights 1 (Analogue)	-	10
Output	Bool	56	E	Roller Shades (Down)	-	10
Output	Float	56	F	Lights 2 (Analogue)	-	120
Output	Bool	57	E	Heater	-	1750
Output	Bool	58	E	Siren	-	0
Output	Bool	59	E	Alarm Key Pad (Arm)	-	0
Output	Bool	60	E	Alarm Key Pad (Disarm)	-	0
Output	Float	66	G	Lights (Analogue)	-	50
Output	Float	67	G	Heater 1 (Analogue)	-	750
Output	Bool	68	F	Lights 1	-	10
Output	Float	68	G	Heater 2 (Analogue)	-	750
Output	Bool	69	F	Lights 2	-	120
Output	Bool	70	F	Roller Shades (Up)	-	10
Output	Bool	71	F	Roller Shades (Down)	-	10
Output	Bool	72	F	Garage Door (Open)	-	1000
Output	Bool	73	F	Garage Door (Close)	-	1000
Output	Float	78	H	Lights (Analogue)	-	40
Output	Float	79	H	Heater (Analogue)	-	1000
Output	Bool	83	G	Lights	-	50
Output	Bool	84	G	Roller Shades (Up)	-	10
Output	Bool	85	G	Roller Shades (Down)	-	10

Memory Type	Data Type	Address	Zone	Name	Contact Type	Power
Output	Bool	86	G	Heater 1	-	750
Output	Bool	87	G	Heater 2	-	750
Output	Float	89	I	Lights 1 (Analogue)	-	10
Output	Float	90	I	Lights 2 (Analogue)	-	10
Output	Float	91	I	Heater (Analogue)	-	750
Output	Bool	97	H	Lights	-	40
Output	Bool	98	H	Roller Shades (Up)	-	10
Output	Bool	99	H	Roller Shades (Down)	-	10
Output	Bool	100	H	Heater	-	1000
Output	Float	101	J	Lights (Analogue)	-	40
Output	Float	102	J	Heater (Analogue)	-	750
Output	Bool	110	I	Lights 1	-	10
Output	Bool	111	I	Lights 2	-	10
Output	Bool	112	I	Heater	-	750
Output	Float	112	K	Lights (Analogue)	-	20
Output	Float	113	K	Heater (Analogue)	-	1000
Output	Bool	122	J	Lights	-	40
Output	Bool	123	J	Roller Shades (Up)	-	10
Output	Float	123	L	Lights (Analogue)	-	40
Output	Bool	124	J	Roller Shades (Down)	-	10
Output	Float	124	L	Heater (Analogue)	-	750
Output	Bool	125	J	Heater	-	750

Memory Type	Data Type	Address	Zone	Name	Contact Type	Power
Output	Float	134	M	Lights (Analogue)	-	10
Output	Bool	135	K	Lights	-	20
Output	Float	135	M	Heater (Analogue)	-	500
Output	Bool	136	K	Heater	-	1000
Output	Float	145	N	Lights 1 (Analogue)	-	90
Output	Bool	146	L	Lights	-	40
Output	Float	146	N	Lights 2 (Analogue)	-	30
Output	Bool	147	L	Roller Shades (Up)	-	10
Output	Float	147	N	Lights 3 (Analogue)	-	30
Output	Bool	148	L	Roller Shades (Down)	-	10
Output	Float	148	N	Heater (Analogue)	-	1500
Output	Bool	149	L	Heater	-	750
Output	Float	158	O	Lights Porch 1 (Analogue)	-	60
Output	Bool	159	M	Lights	-	10
Output	Float	159	O	Lights Porch 2 (Analogue)	-	80
Output	Bool	160	M	Roller Shades (Up)	-	10
Output	Float	160	O	Lights Pool (Analogue)	-	160
Output	Bool	161	M	Roller Shades (Down)	-	10
Output	Float	161	O	Lights Garden (Analogue)	-	40
Output	Bool	162	M	Heater	-	500
Output	Float	162	O	Lights Entrance (Analogue)	-	10
Output	Bool	172	N	Lights 1	-	90
Output	Bool	173	N	Lights 2	-	30

Memory Type	Data Type	Address	Zone	Name	Contact Type	Power
Output	Bool	174	N	Lights 3	-	30
Output	Bool	175	N	Roller Shades (Up)	-	10
Output	Bool	176	N	Roller Shades (Down)	-	10
Output	Bool	177	N	Heater	-	1500
Output	Bool	187	O	Lights Porch 1	-	60
Output	Bool	188	O	Lights Porch 2	-	80
Output	Bool	189	O	Lights Pool	-	160
Output	Bool	190	O	Lights Garden	-	40
Output	Bool	191	O	Lights Entrance	-	10
Output	Bool	192	O	Siren	-	0
Output	Bool	193	O	Entrance Gate (Open)	-	1000
Output	Bool	194	O	Entrance Gate (Close)	-	1000
Memory	Bool	256	-	Using DLST	-	0
Memory	Float	129	-	Time Scalev	-	0
Memory	Float	130	-	Latitude	-	0
Memory	Float	131	-	Longitude	-	0
Memory	Float	132	-	Air Temperature [K]	-	0
Memory	Float	133	-	Relative Humidity	-	0
Memory	Float	134	-	Minimum Air Temperature [K]	-	0
Memory	Float	135	-	Maximum Air Temperature [K]	-	0
Memory	Float	136	-	Dew Point Temperature [K]	-	0
Memory	Float	137	-	Wind Speed [m/s]	-	0

Memory Type	Data Type	Address	Zone	Name	Contact Type	Power
Memory	Float	138	-	Cloudiness	-	0
Memory	Float	139	-	Sun Zenith	-	0
Memory	Float	140	-	Sun Azimuth	-	0
Memory	Float	141	-	Instant Power [w]	-	0
Memory	Float	142	-	Current Hour Energy Consumption [J]	-	0
Memory	Float	143	-	Current Day Energy Consumption [J]	-	0
Memory	Float	144	-	Current Week Energy Consumption [J]	-	0
Memory	Float	145	-	Current Month Energy Consumption [J]	-	0
Memory	Float	146	-	Last Hour Energy Consumption [J]	-	0
Memory	Float	147	-	Last Day Energy Consumption [J]	-	0
Memory	Float	148	-	Last Week Energy Consumption [J]	-	0
Memory	Float	149	-	Last Month Energy Consumption [J]	-	0
Memory	Float	150	A	Temperature [K]	-	0
Memory	Float	151	B	Temperature [K]	-	0
Memory	Float	152	C	Temperature [K]	-	0
Memory	Float	153	D	Temperature [K]	-	0
Memory	Float	154	E	Temperature [K]	-	0
Memory	Float	155	F	Temperature [K]	-	0
Memory	Float	156	G	Temperature [K]	-	0
Memory	Float	157	H	Temperature [K]	-	0
Memory	Float	158	I	Temperature [K]	-	0
Memory	Float	159	J	Temperature [K]	-	0
Memory	Float	160	K	Temperature [K]	-	0

Memory Type	Data Type	Address	Zone	Name	Contact Type	Power
Memory	Float	161	L	Temperature [K]	-	0
Memory	Float	162	M	Temperature [K]	-	0
Memory	Float	163	N	Temperature [K]	-	0
Memory	DateTime	65	-	Date And Time	-	0

ANEXO B: CÓDIGO COMPLETO DE LA CLASE HOMEIO DESARROLLADA EN MATLAB

```
classdef HomeIO < handle
    %HOME I/O Communications class
    %   Grants simple access to all inputs, outputs and memories
    %   This class descends from handle class or either it will not be
    %   capable of processing some things correctly

    % PLEASE NOTE: HomeIO object MUST descend from handle class!
    % https://www.mathworks.com/matlabcentral/answers/183246-updating-property-of-an-
object-without-creating-new-object

    properties

        % Raw data as per the Excel spreadsheet, plus columns
        BaseData;

        % Will hold full data with values
        FullData;

        % Will hold full Inputs, Outputs, Memories
        % and Bools, Floats, DateTimes
        Types = struct();
        %Inputs;
        %Outputs;
        %Memories;
        %Bools;
        %Floats;
        %DateTimes;

        %InputBools;
        %InputFloats;
        %InputDateTimes;
        %OutputBools;
        %OutputFloats;
        %OutputDateTimes;
        %MemoryBools;
        %MemoryFloats;
        %MemoryDateTimes;

        % Will hold all data for a specified Zone (room)
        Zones = struct();
        %ZoneA;
        %ZoneB;
        %ZoneC;
        %ZoneD;
        %ZoneE;
        %ZoneF;
        %ZoneG;
        %ZoneH;
        %ZoneI;
        %ZoneJ;
        %ZoneK;
        %ZoneL;
```

```
%ZoneM;
%ZoneN;
%ZoneO;
%ZoneNone;

% Will Hold devices akin
Devices = struct();

% Inputs category
%LightSwitches;
%UpDownSwitches;
%LightDimmersUpDown;
%DoorDetectors;
%MotionDetectors;
%BrightnessSensorsBool;
%SmokeDetectors;
%AlarmKeyPadArmed;
%GarageDoorSensors;
%EntranceGateSensors;
%RemoteButtons;
%BrightnessSensorsFloat;
%ThermostatTemperatures;
%ThermostatSetPoints;
%RollerShades;
%DateAndTimeInput;

% Outputs category
%LightsBool;
%LightsFloat;
%RollerShadesUpDown;
%HeatersFloat;
%HeatersBool;
%Sirens;
%AlarmKeyPad;
%GarageDoorOpenClose;
%EntranceGate;

% Memories category
%DLST;
%TimeScale;
%LatLongitude;
%EnvironmentValues;
%InstantPower;
%CurrentPower;
%LastPower;
%ZoneTemperatures;
%DateAndTimeMemory;

% Other special, possibly useful classifications
Special = struct()
%InputsNC;
%InputsNO;
%Inputs10V;
%Outputs10V;
%ConflictInputs;
%ConflictOutputs;
%BoolFloatOutputs;

% Use this methods' checkMember and mustMember to check for row membership
```

```
        % --------------------------------
        % Communication items: Listener handler objects and a timer object
        Comm = struct();
        %lh1;
        %lh2;
        %lh3;
        %t; %timer

        % Configs, save just in case
        Config = struct();
        % engineio_path
        % data_path
        % set_timer
        % set_listener

    end

    methods
        function obj = HomeIO(varargin)
            %HOMEIO Constructor
            %   Prepares the class by running the .NET dependency and
            %   updating the MemoryMap once if Home I/O is running
            %   Also allows specifying a non-default path for finding the
            %   Engine I/O DLL library and capability to update data via a
            %   timer or externally


            %% Parse inputs
            default_engineio_path = strcat(pwd,'\EngineIO.dll');
            default_data_path     = strcat(pwd,'\homeio_full.xlsx');
            default_timer         = false;
            default_listen        = true;

            p = inputParser;
            p.addParameter('engineio_path',default_engineio_path,@mustBeFile);
            p.addParameter('data_path',default_data_path,@mustBeFile);
            p.addParameter('set_timer',default_timer,@mustBeNumericOrLogical);
            p.addParameter('set_listener',default_listen,@mustBeNumericOrLogical);
            p.parse(varargin{:});

            obj.Config.engineio_path = p.Results.engineio_path;
            obj.Config.data_path     = p.Results.data_path;
            obj.Config.set_timer     = p.Results.set_timer;
            obj.Config.set_listener  = p.Results.set_listener;

            %% Prepare required libraries and data
            % Link Engine I/O library now (ensures the correct path)
            obj.connectHomeIO(obj.Config.engineio_path);

            % Read and prepare all data from the table (same with path)
            obj.BaseData = obj.readExcelData(obj.Config.data_path);

            % Adapt power consumption to work with the estimatePower method
            % for outputs that are in range 0-10V
            obj.changePower10V();

            % Split data into fixed groups
            obj.splitData();

            % Add a column to store values, DateTime values must be stored
```

```
            % in numerical form (if ever rendered usable). In another
            % variable to preserve table data structure
            obj.FullData = obj.BaseData;
            obj.FullData.Value = nan(height(obj.FullData),1);

            % Check Home I/O, update, preload values
            % Just warn, in case we are using an offline object
            if obj.checkHomeIO("action","warning")
                obj.updateHomeIO();
                obj.FullData.Value = obj.readValues(obj.BaseData);

                % Read again after a couple frames:
                % 1st read always goes wrong next ones go right
                pause(2/60)
                obj.updateHomeIO();
                obj.FullData.Value = obj.readValues(obj.BaseData);

            end

            %% Event listeners and timer setup

            % Listeners if specified
            if obj.Config.set_listener
                obj.Comm.lh1 =
listener(EngineIO.MemoryMap.Instance,'InputsValueChanged',@obj.OnValuesChanged);
                obj.Comm.lh2 =
listener(EngineIO.MemoryMap.Instance,'OutputsValueChanged',@obj.OnValuesChanged);
                obj.Comm.lh3 =
listener(EngineIO.MemoryMap.Instance,'MemoriesValueChanged',@obj.OnValuesChanged);
            end

            % Timer if specified
            if obj.Config.set_timer
                obj.Comm.t = timer;
                obj.Comm.t.BusyMode = 'drop'; % maybe 'queue' could fit other needs
                obj.Comm.t.ExecutionMode = 'fixedRate';
                %obj.Comm.t.TasksToExecute = 10; % test
                %obj.Comm.t.Period = 1/60;
                obj.Comm.t.Period = 1;

                if obj.Config.set_listener
                    obj.Comm.t.TimerFcn = @obj.OnTimerCall;
                    %obj.Comm.t.StopFcn  = @obj.OnTimerEnd;
                    %obj.Comm.t.ErrorFcn = @obj.OnTimerEnd;
                else
                    obj.Comm.t.TimerFcn = @obj.OnTimerCallNoListener;
                    %obj.Comm.t.StopFcn  = @obj.OnTimerEnd;
                    %obj.Comm.t.ErrorFcn = @obj.OnTimerEnd;
                end

                tic; % for performance measurements
                start(obj.Comm.t);
            end
        end

        function out = checkHomeIO(obj,varargin)
            %checkHomeIO checks if Home I/O is running and warns or errors
            %   if appropriate value is supplied

            %% Parse inputs
```

```
            default_action = 'none';

            p = inputParser;
            p.addParameter('action',default_action,@mustBeText);
            p.parse(varargin{:});

            action = p.Results.action;
            mustBeMember(action,{'warning','error','none'});

            % https://stackoverflow.com/questions/858301/detect-matlab-processes-
from-within-matlab
            % Wolfie/Edric's answers edited to check Home IO process
            %[~,w] = dos( 'tasklist /fi "IMAGENAME eq Home IO.exe" /fo "csv"' );
            %out = length( regexp( w, '"Home IO.exe"' ) );

            % Faster approach with included .NET object
            %https://www.mathworks.com/matlabcentral/answers/40617-how-to-find-the-
process-id-pid-in-matlab
            out = System.Diagnostics.Process.GetProcessesByName("Home IO").Length;

            if action ~= "none" && out ~=1
                if action == "warning"
                    warning('Unable to find Home I/O process or Home I/O running more
than once. Results might be inaccurate.')
                    out = 0;
                elseif action == "error"
                    obj.delete()
                    error('Unable to find Home I/O process or Home I/O running more
than once. Stopping...')
                end
            end
        end

        function values = readValues(obj,varargin)
            %Reads values from the supplied array in the argument. If no
            %   argument is provided, reads memories from every line
            %   included in the base data property (obj.BaseData)

            %% Parse inputs
            default_data = obj.BaseData;

            p = inputParser;
            p.addOptional('data',default_data); % Must be member of BaseData
            p.parse(varargin{:});

            data = p.Results.data;
            obj.mustMember(data,obj.BaseData);

            %% Work with data

            % Preset values with NaN of correct size
            values = nan(height(data),1);

            % Attempt at speeding up more: switch-case method
            for i=1:height(data)
                switch data.VarType(i)
                    case 1
                        values(i) =
EngineIO.MemoryMap.Instance.GetBit(data.Address(i), EngineIO.MemoryType.Input).Value;
                    case 2
```

```
                        values(i) =
EngineIO.MemoryMap.Instance.GetFloat(data.Address(i),
EngineIO.MemoryType.Input).Value;
                    case 3
                        % Datenum accepts double values only
                        temp =
EngineIO.MemoryMap.Instance.GetDateTime(data.Address(i),
EngineIO.MemoryType.Input).Value;
                        temp = double([temp.Year temp.Month temp.Day temp.Hour
temp.Minute temp.Second+temp.Millisecond/1000]);
                        values(i) = datenum(temp(1), temp(2), temp(3), temp(4),
temp(5), temp(6));

                    case 4
                        values(i) =
EngineIO.MemoryMap.Instance.GetBit(data.Address(i),
EngineIO.MemoryType.Output).Value;
                    case 5
                        values(i) =
EngineIO.MemoryMap.Instance.GetFloat(data.Address(i),
EngineIO.MemoryType.Output).Value;
                    case 6 % Does never happen but to keep code consistency
                        % Datenum accepts double values only
                        temp =
EngineIO.MemoryMap.Instance.GetDateTime(data.Address(i),
EngineIO.MemoryType.Output).Value;
                        temp = double([temp.Year temp.Month temp.Day temp.Hour
temp.Minute temp.Second+temp.Millisecond/1000]);
                        values(i) = datenum(temp(1), temp(2), temp(3), temp(4),
temp(5), temp(6));

                    case 7
                        values(i) =
EngineIO.MemoryMap.Instance.GetBit(data.Address(i),
EngineIO.MemoryType.Memory).Value;
                    case 8
                        values(i) =
EngineIO.MemoryMap.Instance.GetFloat(data.Address(i),
EngineIO.MemoryType.Memory).Value;
                    case 9
                        % Datenum accepts double values only
                        temp =
EngineIO.MemoryMap.Instance.GetDateTime(data.Address(i),
EngineIO.MemoryType.Memory).Value;
                        temp = double([temp.Year temp.Month temp.Day temp.Hour
temp.Minute temp.Second+temp.Millisecond/1000]);
                        values(i) = datenum(temp(1), temp(2), temp(3), temp(4),
temp(5), temp(6));
                    otherwise
                        error("Unrecognized case: got an unknown value at data row
%i",i);
                end
            end
            % returns values
        end

        function out = getRows(obj,varargin)
            %getRows fetches appropriate rows from the obj.FullData
            %   property. If no data array is supplied, returns the full
            %   table of obj.FullData
```

```
        %% Parse inputs
        default_data = obj.BaseData;

        p = inputParser;
        p.addOptional('data',default_data); % Must be member of BaseData
        p.parse(varargin{:});

        data = p.Results.data;
        mustBeMember(data,obj.BaseData);

        %% Select and return rows from obj.FullData
        out = obj.getRowsFromRowIDs(data.RowID);

    end

    function out = getRowsFromRowIDs(obj,rows)
        p = inputParser;
        p.addRequired('rows',@mustBeNumeric);
        p.parse(rows);

        out = obj.FullData(rows,:);
    end

    function out = getValues(obj,varargin)
        %getValues fetches current values (only) from the rows of the
        %   obj.FullData property. If no data array is supplied,
        %   returns the full values table of obj.FullData

        %% Parse inputs
        default_data = obj.BaseData;

        p = inputParser;
        p.addOptional('data',default_data); % Must be member of BaseData
        p.parse(varargin{:});

        data = p.Results.data;
        mustBeMember(data,obj.BaseData);

        %% Select and return values from obj.FullData
        out = obj.getValuesFromRowIDs(data.RowID);

    end

    function out = getValuesFromRowIDs(obj,rows)
        p = inputParser;
        p.addRequired('rows',@mustBeNumeric);
        p.parse(rows);

        out = obj.FullData.Value(rows);
    end

    function setValues(obj,data,values,varargin)
        %setValues sets values to specified data. Only works for Output
        %   values as these are the only settable (gives a warning
        %   otherwise).
        %   Required parameters: data and values;
        %   Optional parameters: CheckConflicts deactivates conflicting
        %   outputs (for example open and close garage door, keeps the
        %   last one); CapValues caps Bool values to integer 0-1 and
```

```
            %   Float values that have 0-10 limits

            %% Parse inputs

            def_conflicts   = true;
            def_capvalues   = true;
            def_update      = true;
            def_checkHomeIO = true;

            p = inputParser;
            p.addRequired("data")  % Unable to find verif. function
            p.addRequired("values") % Same
            p.addParameter("checkConflicts",def_conflicts,@mustBeNumericOrLogical)
            p.addParameter("capValues",def_capvalues,@mustBeNumericOrLogical)
            p.addParameter("update",def_update,@mustBeNumericOrLogical)
            p.addParameter("checkHomeIO",def_checkHomeIO,@mustBeNumericOrLogical)

            p.parse(data,values,varargin{:});

            data           = p.Results.data;
            values         = p.Results.values;
            checkConflicts = p.Results.checkConflicts;
            capValues      = p.Results.capValues;
            update         = p.Results.update;
            checkHomeIO    = p.Results.checkHomeIO;

            %% Check HomeIO since this petition can come from outside,
            % also check and sanitize other variable issues
            if checkHomeIO
                obj.checkHomeIO("action","error");
            end

            obj.mustMember(data,obj.Types.Outputs);
            mustBeNumericOrLogical(values); % NaNs get hunted later

            % Reject if no coincidence in heights
            n = height(data);
            if height(values) ~= n
                error("Height of data (%d) does not match with height of values
provided (%d)",n,height(values))
            end

            %% Process values
            % Prepare checks for membership for all the data array in a
            % timely manner
            if capValues
                % Just a logical value works
                capRows = obj.checkMember(data,obj.Special.Outputs10V);
            end
            if checkConflicts
                % We need the row value here
                [~,conflictRows] = obj.checkMember(data,obj.Special.ConflictOutputs);
                [~,BoolFloatRows] =
obj.checkMember(data,obj.Special.BoolFloatOutputs);
            end

            % Get the home devices and set the values in different steps.

            for i=1:n
                % Get device type and address to temp var (no table)
```

```
                % Addresses must be extracted from the table too
                type = data.DataType(i); % "Bool", "Float", no "DateTime"
                addr = data.Address(i);

                if type == "Bool"
                    device = EngineIO.MemoryMap.Instance.GetBit(addr,
EngineIO.MemoryType.Output);
                elseif type == "Float"
                    device = EngineIO.MemoryMap.Instance.GetFloat(addr,
EngineIO.MemoryType.Output);
                end

                % Use the capValues and checkConflicts if asked
                if capValues
                    if type == "Bool"
                        if values(i)
                            values(i) = 1;
                        else
                            values(i) = 0;
                        end
                    elseif type == "Float"
                        if capRows(i)
                            if values(i) > 10
                                values(i) = 10;
                            elseif values(i) < 0
                                values(i) = 0;
                                %  else % Not needed
                                %  values(i) = values(i);
                            end
                        end
                    else % There are no output DateTimes, throw an error
                        error("Unrecognized output type %s for data address
%d",type,addr);
                    end
                end

                % Odd row: Disable next value
                % Even row: Disable prev. value
                if checkConflicts
                    % Incompatible bool values
                    if conflictRows(i)
                        if rem(conflictRows(i),2) % == 1
                            addr2 =
obj.Special.ConflictOutputs.Address(conflictRows(i)+1);
                            %addr2 = addr2{:,:};
                            device2 = EngineIO.MemoryMap.Instance.GetBit(addr2,
EngineIO.MemoryType.Output);
                        else % rem(conflictRows(i),2) == 0
                            addr2 =
obj.Special.ConflictOutputs.Address(conflictRows(i)-1);
                            %addr2 = addr2{:,:};
                            device2 = EngineIO.MemoryMap.Instance.GetBit(addr2,
EngineIO.MemoryType.Output);
                        end
                        device2.Value = 0;
                    end

                    % Incompatible Bool/Float values
                    if BoolFloatRows(i)
```

```
                        if rem(BoolFloatRows(i),2) % == 1, it's a Bool so the other
one is Float
                            addr2 =
obj.Special.BoolFloatOutputs.Address(BoolFloatRows(i)+1);
                            device2 = EngineIO.MemoryMap.Instance.GetFloat(addr2,
EngineIO.MemoryType.Output);
                        else % rem(idx,2) == 0, , it's a Float so the other one is
Bool
                            addr2 =
obj.Special.BoolFloatOutputs.Address(BoolFloatRows(i)-1);
                            device2 = EngineIO.MemoryMap.Instance.GetBit(addr2,
EngineIO.MemoryType.Output);
                        end
                        device2.Value = 0;
                    end

                end

                % Set all values now
                device.Value = values(i);
            end

            % Sometimes all other values update correctly except for these
            % Give our model a hand:
            obj.FullData.Value(data.RowID) = values;

            %% Force update if required; values will be read by the events
            if update
                obj.updateHomeIO();
            end
        end

        % Not providing setValuesFromRowID because necessarily goes back
        % to full row data to get DataType, MemoryType and Address...

        function out = estimatePower(obj)
            %estimatePower returns the power consumption estimated via the
            % devices currently active. Does not need Home IO running.
            out = sum(obj.FullData.Power.*obj.FullData.Value);
        end

        function OnTimerCall(obj,~,~)
            %Timer callback if event handlers are available

            %profile resume
            %tim = toc;
            obj.checkHomeIO("action","error");
            obj.updateHomeIO(); % Callback is automatically executed by listener
handles
            %disp(toc-tim);
            %profile off
        end

        function OnTimerCallNoListener(obj,~,~)
            %Timer Callback if listener handles are deactivated. Update is
            %   done manually via the readValues method.

            %profile resume
            %tim = toc;
            obj.checkHomeIO("action","error");
```

```
            obj.updateHomeIO();
            obj.FullData.Value = obj.readValues();
            %disp(toc-tim);
            %profile off
        end

        function delete(obj)
            %Remove timers and listeners to avoid unnecessary cluttering
            if obj.Config.set_timer
                stop(obj.Comm.t);
                delete(obj.Comm.t);
            end

            if obj.Config.set_listener
                delete(obj.Comm.lh1);
                delete(obj.Comm.lh2);
                delete(obj.Comm.lh3);
            end
            EngineIO.MemoryMap.Instance.Dispose();
        end

        function s = saveobj(obj)
            %Save our object to a class to reconstruct it. Configuration
            % parameters are just enough to accomplish this task.
            s.Config = obj.Config;
        end

    end

    methods (Access = private)

        function changePower10V(obj)
            %changePower10V divides by 10 these outputs that range in the
            %value 0-10V, so that they will consume power proportional to
            %that voltage
            % All of these are Output Floats, VarType == 5
            rows = obj.BaseData.VarType == 5;
            obj.BaseData.Power(rows) = obj.BaseData.Power(rows)/10;
        end

        function OnValuesChanged(obj,varargin)
            % Callback function called by event handlers after updating
            % Home I/O from the update function.

            % varargin{1} is the name of the event, which is useless
            % varargin{2}.MemoriesBit, varargin{2}.MemoriesFloat,
varargin{2}.MemoriesDateTime
            % These three are vectors, length specified in
varargin{2}.Memories[[TYPE]].Length
            % Direct access, eg, by using varargin(2).MemoriesFloat(i)

            obj.updateFromEvent(varargin{2});
        end

        function updateFromEvent(obj,data)
            %updateFromEvent updates values from the obj.FullData attribute
            %   caught from the events.

            % Get MemoriesBit values
            if data.MemoriesBit.Length > 0
```

```
                for i=1:data.MemoriesBit.Length
                    % Get data, then get the index
                    datarow = data.MemoriesBit(i);

                    % Below is equivalent to these 2 lines, but faster
                    %rowid = obj.Types.Bools.RowID( ...
                    %    obj.Types.Bools.MemoryType == string(datarow.MemoryType) &
...
                    %    obj.Types.Bools.Address == datarow.Address);
                    %obj.FullData.Value(rowid) = datarow.Value;

                    obj.FullData.Value( obj.Types.Bools.RowID( ...
                        obj.Types.Bools.MemoryType == string(datarow.MemoryType) &
...
                        obj.Types.Bools.Address == datarow.Address) ...
                        ) = datarow.Value;

                end
            end

            if data.MemoriesFloat.Length > 0
                for i=1:data.MemoriesFloat.Length
                    % Get data, then get the index
                    datarow = data.MemoriesFloat(i);

                    % Below is equivalent to these 2 lines, but faster
                    %rowid = obj.Types.Floats.RowID( ...
                    %    obj.Types.Floats.MemoryType == string(datarow.MemoryType) &
...
                    %    obj.Types.Floats.Address == datarow.Address);
                    %obj.FullData.Value(rowid) = datarow.Value;

                    obj.FullData.Value(obj.Types.Floats.RowID( ...
                        obj.Types.Floats.MemoryType == string(datarow.MemoryType) &
...
                        obj.Types.Floats.Address == datarow.Address) ...
                        ) = datarow.Value;

                end
            end

            if data.MemoriesDateTime.Length > 0
                for i=1:data.MemoriesDateTime.Length
                    % Get data, then get the index
                    datarow = data.MemoriesDateTime(i);

                    % Datenum accepts double values only
                    temp = datarow.Value;
                    temp = double([temp.Year temp.Month temp.Day temp.Hour
temp.Minute temp.Second+temp.Millisecond/1000]);

                    obj.FullData.Value(obj.Types.DateTimes.RowID( ...
                        obj.Types.DateTimes.MemoryType == string(datarow.MemoryType)
& ...
                        obj.Types.DateTimes.Address == datarow.Address) ...
                        ) = datenum(temp(1), temp(2), temp(3), temp(4), temp(5),
temp(6));
                end
            end
        end
```

```
        function splitData(obj)
            % splitData splits the data in zones (rooms) and other logical
            % categories, to enable ease of usage by the end user.

            %% Split per useful categories

            % Memory Types
            obj.Types.Inputs   = obj.BaseData(obj.BaseData.MemoryType == 'Input',:);
            obj.Types.Outputs  = obj.BaseData(obj.BaseData.MemoryType == 'Output',:);
            obj.Types.Memories = obj.BaseData(obj.BaseData.MemoryType == 'Memory',:);

            obj.Types.Bools     = obj.BaseData(obj.BaseData.DataType == 'Bool',:);
            obj.Types.Floats    = obj.BaseData(obj.BaseData.DataType == 'Float',:);
            obj.Types.DateTimes = obj.BaseData(obj.BaseData.DataType ==
'DateTime',:);

            obj.Types.InputBools      = obj.BaseData(obj.BaseData.VarType == 1,:);
            obj.Types.InputFloats     = obj.BaseData(obj.BaseData.VarType == 2,:);
            obj.Types.InputDateTimes  = obj.BaseData(obj.BaseData.VarType == 3,:);
            obj.Types.OutputBools     = obj.BaseData(obj.BaseData.VarType == 4,:);
            obj.Types.OutputFloats    = obj.BaseData(obj.BaseData.VarType == 5,:);
            obj.Types.OutputDateTimes = obj.BaseData(obj.BaseData.VarType == 6,:);
            obj.Types.MemoryBools     = obj.BaseData(obj.BaseData.VarType == 7,:);
            obj.Types.MemoryFloats    = obj.BaseData(obj.BaseData.VarType == 8,:);
            obj.Types.MemoryDateTimes = obj.BaseData(obj.BaseData.VarType == 9,:);

            % Zones
            obj.Zones.ZoneA    = obj.BaseData(obj.BaseData.Zone == 'A',:);
            obj.Zones.ZoneB    = obj.BaseData(obj.BaseData.Zone == 'B',:);
            obj.Zones.ZoneC    = obj.BaseData(obj.BaseData.Zone == 'C',:);
            obj.Zones.ZoneD    = obj.BaseData(obj.BaseData.Zone == 'D',:);
            obj.Zones.ZoneE    = obj.BaseData(obj.BaseData.Zone == 'E',:);
            obj.Zones.ZoneF    = obj.BaseData(obj.BaseData.Zone == 'F',:);
            obj.Zones.ZoneG    = obj.BaseData(obj.BaseData.Zone == 'G',:);
            obj.Zones.ZoneH    = obj.BaseData(obj.BaseData.Zone == 'H',:);
            obj.Zones.ZoneI    = obj.BaseData(obj.BaseData.Zone == 'I',:);
            obj.Zones.ZoneJ    = obj.BaseData(obj.BaseData.Zone == 'J',:);
            obj.Zones.ZoneK    = obj.BaseData(obj.BaseData.Zone == 'K',:);
            obj.Zones.ZoneL    = obj.BaseData(obj.BaseData.Zone == 'L',:);
            obj.Zones.ZoneM    = obj.BaseData(obj.BaseData.Zone == 'M',:);
            obj.Zones.ZoneN    = obj.BaseData(obj.BaseData.Zone == 'N',:);
            obj.Zones.ZoneO    = obj.BaseData(obj.BaseData.Zone == 'O',:);
            obj.Zones.ZoneNone = obj.BaseData(obj.BaseData.Zone == '-',:);

            % By object types
            % Input Bools
            obj.Devices.LightSwitches         =
obj.Types.InputBools(contains(obj.Types.InputBools.Name,"Light Switch") &
~contains(obj.Types.InputBools.Name,"Dimmer"),:);
            obj.Devices.UpDownSwitches        =
obj.Types.InputBools(contains(obj.Types.InputBools.Name,"Up/Down Switch"),:);
            obj.Devices.LightDimmersUpDown    =
obj.Types.InputBools(contains(obj.Types.InputBools.Name,"Light Switch Dimmer"),:);
            obj.Devices.DoorDetectors         =
obj.Types.InputBools(contains(obj.Types.InputBools.Name,"Door Detector"),:);
            obj.Devices.MotionDetectors       =
obj.Types.InputBools(contains(obj.Types.InputBools.Name,"Motion Detector"),:);
            obj.Devices.BrightnessSensorsBool =
obj.Types.InputBools(contains(obj.Types.InputBools.Name,"Brightness Sensor"),:);
```

```
            obj.Devices.SmokeDetectors        =
obj.Types.InputBools(contains(obj.Types.InputBools.Name,"Smoke Detector"),:);
            obj.Devices.AlarmKeyPadArmed      =
obj.Types.InputBools(contains(obj.Types.InputBools.Name,"Alarm Key Pad"),:);
            obj.Devices.GarageDoorSensors     =
obj.Types.InputBools(contains(obj.Types.InputBools.Name,"Garage Door"),:);
            obj.Devices.EntranceGateSensors   =
obj.Types.InputBools(contains(obj.Types.InputBools.Name,"Entrance Gate"),:);
            obj.Devices.RemoteButtons         =
obj.Types.InputBools(contains(obj.Types.InputBools.Name,"Remote Button"),:);

            % Input Floats
            obj.Devices.BrightnessSensorsFloat =
obj.Types.InputFloats(contains(obj.Types.InputFloats.Name,"Brightness Sensor"),:);
            obj.Devices.ThermostatTemperatures =
obj.Types.InputFloats(contains(obj.Types.InputFloats.Name,"(Room Temperature)"),:);
            obj.Devices.ThermostatSetPoints    =
obj.Types.InputFloats(contains(obj.Types.InputFloats.Name,"(Set Point)"),:);
            obj.Devices.RollerShades           =
obj.Types.InputFloats(contains(obj.Types.InputFloats.Name,"Roller Shades"),:);

            % Input DateTime
            obj.Devices.DateAndTimeInput =
obj.Types.InputDateTimes(contains(obj.Types.InputDateTimes.Name,"Date and Time"),:);

            % Output Bools
            obj.Devices.LightsBool          =
obj.Types.OutputBools(contains(obj.Types.OutputBools.Name,"Lights"),:);
            obj.Devices.RollerShadesUpDown  =
obj.Types.OutputBools(contains(obj.Types.OutputBools.Name,"Roller Shades"),:);
            obj.Devices.HeatersBool         =
obj.Types.OutputBools(contains(obj.Types.OutputBools.Name,"Heater"),:);
            obj.Devices.Sirens              =
obj.Types.OutputBools(contains(obj.Types.OutputBools.Name,"Siren"),:);
            obj.Devices.AlarmKeyPad         =
obj.Types.OutputBools(contains(obj.Types.OutputBools.Name,"Alarm Key Pad"),:);
            obj.Devices.GarageDoorOpenClose =
obj.Types.OutputBools(contains(obj.Types.OutputBools.Name,"Garage Door"),:);
            obj.Devices.EntranceGate        =
obj.Types.OutputBools(contains(obj.Types.OutputBools.Name,"Entrance Gate"),:);

            % Output Floats
            obj.Devices.LightsFloat  =
obj.Types.OutputFloats(contains(obj.Types.OutputFloats.Name,"Lights"),:);
            obj.Devices.HeatersFloat =
obj.Types.OutputFloats(contains(obj.Types.OutputFloats.Name,"Heater"),:);

            % Output DateTimes
            % Nonexistent

            % Memory Bools
            obj.Devices.DLST =
obj.Types.MemoryBools(contains(obj.Types.MemoryBools.Name,"DLST"),:);

            % Memory Floats
            obj.Devices.TimeScale         =
obj.Types.MemoryFloats(contains(obj.Types.MemoryFloats.Name,"Time Scale"),:);
            obj.Devices.LatLongitude      =
obj.Types.MemoryFloats(contains(obj.Types.MemoryFloats.Name,["Latitude",
"Longitude"]),:);
```

```
            obj.Devices.EnvironmentValues =
obj.Types.MemoryFloats(obj.Types.MemoryFloats.Address >= 132 &
obj.Types.MemoryFloats.Address <= 140,:);
            obj.Devices.InstantPower      =
obj.Types.MemoryFloats(contains(obj.Types.MemoryFloats.Name,"Instant Power"),:);
            obj.Devices.CurrentPower      =
obj.Types.MemoryFloats(contains(obj.Types.MemoryFloats.Name,"Current"),:);
            obj.Devices.LastPower         =
obj.Types.MemoryFloats(contains(obj.Types.MemoryFloats.Name,"Last"),:);
            obj.Devices.ZoneTemperatures  =
obj.Types.MemoryFloats(obj.Types.MemoryFloats.Zone ~= "-",:);

            % Memory DateTimes
            obj.Devices.DateAndTimeMemory =
obj.Types.MemoryDateTimes(contains(obj.Types.MemoryDateTimes.Name,"Date"),:);


            % Other memberships
            obj.Special.InputsNO        =
obj.Types.InputBools(obj.Types.InputBools.ContactType == 'NO',:);
            obj.Special.InputsNC        =
obj.Types.InputBools(obj.Types.InputBools.ContactType == 'NC',:);
            obj.Special.Inputs10V       = [obj.Devices.BrightnessSensorsFloat;
obj.Devices.RollerShades];
            obj.Special.Outputs10V      = [obj.Devices.LightsFloat;
obj.Devices.HeatersFloat];

            obj.Special.ConflictInputs   = [obj.Devices.UpDownSwitches;
obj.Devices.LightDimmersUpDown];
            obj.Special.ConflictOutputs  = [obj.Devices.RollerShadesUpDown;
obj.Devices.GarageDoorOpenClose; obj.Devices.EntranceGate; obj.Devices.AlarmKeyPad];

            % Special.BoolFloatOutputs will save us time if stored
            % alternating the corresponding Bool and Float consecutively
            % Not pretty but useful. TOO SLOW but only runs once
            obj.Special.BoolFloatOutputs = table();

            for i=1:size(obj.Devices.LightsBool)
                obj.Special.BoolFloatOutputs = [obj.Special.BoolFloatOutputs;
obj.Devices.LightsBool(i,:); obj.Devices.LightsFloat(i,:)];
            end

            for i=1:size(obj.Devices.HeatersBool)
                obj.Special.BoolFloatOutputs = [obj.Special.BoolFloatOutputs;
obj.Devices.HeatersBool(i,:); obj.Devices.HeatersFloat(i,:)];
            end

        end

    end

    methods (Static)

        function connectHomeIO(varargin)
            %Connects this thread to the Engine I/O DLL library

            default_path = strcat(pwd,'\EngineIO.dll');
            p = inputParser;
            p.addOptional('engineio_path',default_path,@mustBeFile);
            p.parse(varargin{:});
```

```
        engineio_path = p.Results.engineio_path;

        % Link Engine I/O library
        NET.addAssembly(engineio_path);
    end

    function out = readExcelData(varargin)
        %Read and preprocess all data from the Excel table to get the
        %   array to assign, typically, to obj.BaseData

        %% Parse inputs
        default_data_path     = strcat(pwd,'\homeio_full.xlsx');

        p = inputParser;
        p.addOptional('data_path',default_data_path,@mustBeFile);
        p.parse(varargin{:});

        data_path = p.Results.data_path;

        %% Read and prepare all data from the table (same with path)
        warning('OFF', 'MATLAB:table:ModifiedAndSavedVarnames')
        out = readtable(data_path,'VariableNamingRule',"modify");
        warning('ON', 'MATLAB:table:ModifiedAndSavedVarnames')

        % Set strings to MemoryType, DataType, Zone and ContactType
        % Makes arrays (supposedly) faster to check than categorical
        out.MemoryType  = string(out.MemoryType);
        out.DataType    = string(out.DataType);
        out.Zone        = string(out.Zone);
        out.ContactType = string(out.ContactType);
        out.Name        = string(out.Name);

        % Add VarType to speed array checks (saves string comparisons)
        % Calculate VarType according to MemoryType and DataType
        % VarType = 1 Input Bool
        % VarType = 2 Input Float
        % VarType = 3 Input DateTime
        % VarType = 4 Output Bool
        % VarType = 5 Output Float
        % VarType = 6 Output DateTime
        % VarType = 7 Memory Bool
        % VarType = 8 Memory Float
        % VarType = 9 Memory DateTime

        out.VarType = ...
            1 * (out.MemoryType == 'Input' & out.DataType == 'Bool') + ...
            2 * (out.MemoryType == 'Input' & out.DataType == 'Float') + ...
            3 * (out.MemoryType == 'Input' & out.DataType == 'DateTime') + ...
            4 * (out.MemoryType == 'Output' & out.DataType == 'Bool') + ...
            5 * (out.MemoryType == 'Output' & out.DataType == 'Float') + ...
            6 * (out.MemoryType == 'Output' & out.DataType == 'DateTime') + ...
            7 * (out.MemoryType == 'Memory' & out.DataType == 'Bool') + ...
            8 * (out.MemoryType == 'Memory' & out.DataType == 'Float') + ...
            9 * (out.MemoryType == 'Memory' & out.DataType == 'DateTime');

        out = movevars(out,'VarType','Before','MemoryType');

        % Sort to keep the like variables near one another
        % Mainly affects Outputs, which are really unordered
```

```
            out = sortrows(out,{'VarType','Address'},'ascend');

            % Add RowID parameter to make searches faster and less cluttery
            % Calculate RowIDs here if applicable
            for i=1:height(out)
                out.RowID(i) = i;
            end

            out = movevars(out,'RowID','Before','VarType');
        end

        function updateHomeIO()
            %Simply runs the Home I/O command to update the MemoryMap
            EngineIO.MemoryMap.Instance.Update();
        end

        function [Lia, Locb] = checkMember(A,B)
            % A wrapper function of ismember for our table properties
            % which speeds up comparisons based on RowIDs instead of
            % checking the full rows and saves a lot of time
            [Lia, Locb] = ismember(A.RowID,B.RowID);
        end

        function mustMember(A,B)
            % A wrapper function of mustBeMember for our table properties
            % which speeds up comparisons based on RowIDs instead of
            % checking the full rows and saves a lot of time
            mustBeMember(A.RowID,B.RowID);
        end

        function obj = loadobj(s)
            %loadobj tries to restore the previous object
            %   No need to test for errors with the isstruct check since
            %   the constructor usually generates listeners, which are
            %   non-copyable and must be restored manually (so call
            %   the constructor back with the same arguments)
            engineio_path = s.Config.engineio_path;
            data_path = s.Config.data_path;
            set_timer = s.Config.set_timer;
            set_listener = s.Config.set_listener;

            obj =
HomeIO('engineio_path',engineio_path,'data_path',data_path,'set_timer',set_timer,'set
_listener',set_listener);
        end

    end
end
```