Proyecto Fin de Grado
Grado en Ingeniería de Tecnologías Industriales

Pasarela basada en Arduino para conectar sensores y actuadores al simulador Home I/O

Autor: Adrián Barrera Ramírez
Tutor: Jesús Iván Maza Alcañiz

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

Sevilla, 2018

Proyecto Fin de Grado
Grado en Ingeniería de Tecnologías Industriales

Pasarela basada en Arduino para conectar sensores y actuadores al simulador Home I/O

Autor:
Adrián Barrera Ramírez

Tutor:
Jesús Iván Maza Alcañiz
Profesor titular

Dep. en Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2018

Proyecto Fin de Grado: Pasarela basada en Arduino para conectar sensores y actuadores al simulador Home I/O

Autor: Adrián Barrera Ramírez

Tutor: Jesús Iván Maza Alcañiz

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

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2018

El Secretario del Tribunal

A mi familia

A mis maestros

A mis compañeros

Agradecimientos

Quisiera dar las gracias a todas aquellas amistades que he hecho durante mi estancia en la Escuela, que no ha sido corta, y que en todo momento han estado a mi lado ofreciéndome su ayuda y trabajando codo con codo: Encinas, Castaño, Dani, Jorge, Carmona, Cobacho, Cañada, Barrera, Diego, Aceituno, Pepe, Telo, Noelia, Calvillo, Ana, Lope, Ali, María Jesús, Celia, Víctor, Fernando, Jaime, etc. A todos mis archienemigos de telecomunicaciones: Juan, Andrés G., María Silvestre, Marina, Ana, Lourdes, etc. A mis compañeros de especialidad, que aunque les conocí más tarde, no quita que les tenga menos cariño: Andrés P., Juandi, Gonzalo, etc. y muchos más que seguramente me he dejado atrás.

A todos y cada uno de los profesores que me han impartido clases y que han aportado su granito de arena para que haya llegado hasta aquí. Mención especial a mi profesor del TFG, Iván Maza, por su paciencia y dedicación en cada correo y tutoría.

Y por supuesto, no podía ser menos, a mis padres y a mi hermana Alba, por su apoyo incondicional y por ser mi pilar fundamental, asesorándome en todo momento en mis decisiones y respetándolas. Porque sin ellos no sería la persona quien soy. Gracias por inculcarme todos vuestros valores y por darme la educación que me habéis dado.

Adrián Barrera Ramírez

Sevilla, 2018

Resumen

El término domótica viene de la unión de las palabras domus (en latín, casa) y autónomo (del griego, "que se gobierna a sí mismo").

Se denomina domótica a aquellos sistemas capaces de automatizar una vivienda o edificación de cualquier tipo, aportando servicios de gestión energética, seguridad, bienestar y comunicación, y que pueden estar integrados por medio de redes interiores o exteriores de comunicación, ya sean cableadas o inalámbricas. Se podría definir como la integración de la tecnología en el diseño inteligente de un recinto cerrado.

Hoy en día cada vez son más las familias, por lo general con un perfil de gran poder adquisitivo, que demandan este tipo de tecnologías en sus hogares y establecimientos. Ante esta demanda son muchas las empresas que deciden dedicarse a la implantación de este tipo de tecnologías.

Conforme avancen los años esta demanda irá incrementándose. Por este motivo, se hace de la domótica un sector muy atractivo con múltiples opciones y en continuo desarrollo adaptándose a las nuevas tecnologías del momento. Estar en la calle, mirar tu Smartphone y tener la posibilidad de manipular la temperatura de tu hogar con vistas a tu próxima llegada, o incluso ver el estado de los sensores que se encargan de la seguridad de tu hogar son situaciones cada vez más comunes y todas ellas contribuyen a la mejora de la calidad y del bienestar en todos los sentidos.

Abstract

The term domotic comes from the union of the words domus (in Latin, house) and autonomous (of the Greek, "that governs to itself").

It is called domotics to the systems capable of automating a home or building of any kind, providing energy management, security, welfare and communication services, and which can be integrated by internal or external communication networks, whether wired or wireless. It could be defined as the integration of technology in the intelligent design of an enclosed space.

Nowadays, all families, in general with a high purchasing power profile, demand this type of technology in their homes and establishments. Given this demand, many companies decide to dedicate themselves to the implementation of this type of technology.

As the years advance this demand will increase. For this reason, home automation is a very attractive sector with multiple options and in continuous development adapting to the new technologies of the moment. Being on the street, looking at your smartphone and having the ability to manipulate the temperature of your home with a view to your next arrival, or even see the status of the sensors that are responsible for the security of your home are increasingly common situations and all of them contribute to the improvement of quality and well-being in all senses.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1 INTRODUCCIÓN

"Engineers like to solve problems. If there are no problems handily available, they will create their own problems."

- Scott Adams -

1.1 Un poco de historia

La domótica se inicia a comienzos de la década del '70, cuando aparecieron los primeros dispositivos de automatización en edificios, a base de prueba piloto. Pero fue en la década del '80 cuando los sistemas integrados se utilizaron a nivel comercial, para luego desarrollarse en el aspecto doméstico de las casas urbanas. Allí es cuando la domótica consigue integrar dos sistemas (el eléctrico y el electrónico) en pos de la comunicación integral de los dispositivos del hogar. El desarrollo de la tecnología informática permite la expansión del sistema, sobre todo en países de vanguardia como Estados Unidos, Alemania y Japón.

Acorde a los cambios, el auge de la informática hogareña permite incorporar en los edificios el Sistema de Cableado Estructurado (SCE) que facilita la conexión de terminales y redes. Así, estos edificios reciben el nombre de "inteligentes", por su automatismo al servicio del propietario. El boom de estos rascacielos de oficinas comerciales fue de gran impacto. La domótica permitía lograr una eficiencia inédita para el servicio de dispositivos.

El primer programa que utilizó la domótica fue el Save. Creado en Estados Unidos en 1984, permite lograr eficiencia y bajo consumo de energía en los sistemas de control de edificios inteligentes.

El programa SAVE funcionaba mediante protocolo X10, gracias al cual se podían realizar determinadas acciones por control remoto. El nacimiento del protocolo X10 también es relevante: aterrizó en 1975 por Pico Electronics (Escocia) gracias a 'Telecontrol'. Fue el primer protocolo para domótica –poco después llegarían los protocolos cerrados- y su "sencillez y accesibilidad" le permitió una elevada penetración tanto en EEUU como en determinados países de Europa.

En nuestros días todavía no ha desparecido el X10, todo lo contrario. Cientos de empresas siguen trabajando con él, aunque el nacimiento de otros protocolos abiertos le haga luchar por mantener su cuota de protagonismo. Este tipo de avances (la automatización y la domótica en general) nacieron y crecieron por la necesidad de encontrar la casa ideal, algo que ya se pretendía mucho antes de que smartphones y tabletas dominaran el panorama.

A pesar de todo ello, actualmente sigue siendo la tecnología más utilizada dentro de la domótica. Al transmitir datos por líneas de baja tensión, la relación costo-beneficio sigue siendo la mejor opción en el mercado.

Implantada desde hace más de treinta años, la domótica ha progresado a gran escala desde que se desarrollaron las redes informáticas de comunicación, ya sea por sistema cableado o vía Wi-Fi. El avance tecnológico vino a suplir las carencias de los comienzos, ya que permite integrar de manera eficiente todos los dispositivos tecnológicos de una casa. Con el fin de la década del ’80 la tecnología de un comienzo, destinada a fines comerciales, comienza a llegar a los hogares.

Irrumpe la era de la TIC (la tecnología de informática y comunicaciones), que posibilita entender una forma más realista de comprender la instalación domótica en casa.
En la actualidad hay una oferta consolidada en torno a los servicios de domótica. Nuevos protocolos permiten un desarrollo que en un principio era impensado. Sistemas de desarrollo 2.0 como el ZigBee permiten conformar un protocolo inalámbrico de comunicación domótica.

Zigbee es el nombre de la especificación de un conjunto de protocolos de alto nivel de comunicación inalámbrica para su utilización con radiodifusión digital de bajo consumo, basada en el estándar IEEE 802.15.4 de redes inalámbricas de área personal (wireless personal area network, WPAN). Su objetivo son las aplicaciones que requieren comunicaciones seguras con baja tasa de envío de datos y maximización de la vida útil de sus baterías.

Al requerir una baja tasa de envío de datos, ZigBee es en la actualidad uno de los protocolos más requeridos para las casas “inteligentes”, ya sea en sensores de movimiento, detectores de humo y otras funciones de seguridad en el hogar.

Con la domótica aplicada a la automatización hogareña se mejora en seguridad, confort y ahorro energético, aspectos muy observados por los poseedores de estos sistemas. La llegada de Internet a gran velocidad provocó un giro favorable para su desarrollo.

Figura 1-1 Distintos elementos domóticos que integran una vivienda

En los últimos años el mercado de ofertas se ha extendido, permitiendo encontrar diversas variantes de equipos domésticos de integración domótica, como es el caso del EIB. European Instalation Bus (EIB) es un bus pensado para su utilización como sistema de gestión en la instalación eléctrica de un edificio. Ayuda a optimizar los distintos sistemas de seguridad y funcionalidad que componen una casa. Este protocolo se utiliza en buena cantidad de casas inteligentes de Europa Central. Ejemplo de ello es: control de sistemas como alumbrado, calefacción, aire acondicionado, persianas, alarmas, etc. Para instalaciones fijas.

¿Quién fue Joel Spira?

En 2015 nos dejaba a los 88 años el inventor de la domótica, un título que no se lo discute nadie. Spira adquirió potentes conocimientos tras enrolarse en la Marina y ser reclutado para crear un sistema basado en ondas de radio que permitiera detectar a los enemigos. Al parecer, la bombilla de la domótica se le encendió tras trabajar en un sistema de fusibles para bombas atómicas. El caso es que se olvidó del segmento militar y en 1961 dio a luz Lutron Electronics, su propia empresa.

A Spira se le atribuye la creación del atenuador basado en estados de ánimo. Realmente, los primeros pasos comerciales de la domótica se basaban en este tipo de tecnología: regulación de la temperatura o la luz de los

edificios de oficinas, a las que solo por este tipo de adelantos se les empezó a conceder la etiqueta de 'inteligentes'. Pero el legado de Spira solo había dado comienzo.

Los reguladores fueron los primeros en llegar, pero el hecho de que el propio Spira continuara trabajando incluso la noche antes de morir –trabajaba juntos a sus ingenieros en sistemas inalambricos para casas inteligentes- demuestra que en domótica hay mucho que inventar cada día. 30 años –o más- después de su implantación la domótica sigue progresando a gran escala, sobre todo gracias a las redes wifi.

1.2 Motivación

Debido al gran avance y auge de las tecnologías que integran un hogar domotico que ha tenido lugar en la última década, este proyecto se ha buscado desarrollarlo en un marco muy actual e innovador. Haciéndolo además de atractivo e interesante de cara al espectador, una línea muy interesante de trabajo para fututos proyectos que continúen el trabajo llevado a cabo en el mismo.

Otro de los motivos por el que el trabajo nos parece adecuado para esta temática es su potencial. Al tratarse de un campo en continuo desarrollo y con tantas variantes nos daba libertad para poder trabajar en una línea de investigación sin necesidad de estar sujeto a restricciones o ningún tipo de condición.

La mayor ventaja y característica del simulador HOME I/O es que rompe con todo lo anterior hasta ahora visto dentro del mundo de los simuladores de tecnología domótica, ya que nos ofrece la posibilidad de visualizar a partir de su interfaz toda la casa inteligente, dándonos total libertad para movernos por ella, interactúando con una gran cantidad de sensores y actuadores, y permitiéndonos en tiempo real comprobar el efecto que tiene lugar por los mismos. El poder trabajar con tal cantidad de sensores y actuadores, ya sea de forma manual o programada observando las interacciones entre los elementos, hacen de HOME I/O una herramienta única en el mercado.

Otro de los puntos fuertes de HOME I/O es su conexión en tiempo real con el software CONNECT I/O. Este otro programa, proporcionado también por real games, es el encargado de la programación mediante nodos asociados a cada uno de los diversos sensores y actuadores. Durante la simulación de diagramas de bloques realizados por nosotros mismos, además de permitir millones de variantes, es importate destacar la sencillez conjugada con su animación grafica, ya que nos facilita en gran medida el descubrir errores o comportamientos indeseados.

Todos los softwares relacionados con la domótica y otros soportes digitales tienen un precio elevado. La justificación del por qué se utilizo Arduino se ha basado en su reducido coste. Arduino, además de ofrecernos muchas posibilidades, nos permite trabajar con el software HOME I/O al igual que otros programas de mayor entidad dentro del mundo domotico. Por ello se decidió trabajar con el, ya que nos ofrece una solución válida y muy competente dentro del mercado por un precio realmente bajo.

Además de todo lo mencionado hasta ahora, cabe destacar también el interés didáctico en el simulador HOME I/O y su conexión directa con CONNECT I/O. Gracias a programas como estos son muchos los institutos, universidades e investigaciones, como puede ser este trabajo, que tan sólo requiriendo de un ordenador pueden conseguir domotizar, en mayor o menor medida, una vivienda de forma muy fiel a la realidad, enriqueciendo los conocimientos del alumnado, y por supuesto preparándolo académicamente de cara a un futuro próximo y a la vida real.

1.3 Objetivos del trabajo fin de grado

El objetivo de este proyecto consiste en construir un soporte real programado mediante una pasarela en Arduino y comunicado con el software de programación CONNECT I/O, permitiéndonos controlar los distintos elementos o tecnologías que encontraremos en nuestro hogar virtual e inteligente a medida que nos desplacemos por él, proporcionado por el software HOME I/O y desarrollado por la compañía portuguesa Real Games.

El proyecto abarcará los siguientes temas:

- Manual sobre cómo realizar la comunicación entre los distintos programas que emplearemos en nuestra computadora: HOME I/O, CONNECT I/O y Arduino. Reconocimiento y configuración del hardware Arduino en CONNECT I/O.
- Automatización de las diversas acciones de las que estará compuesto el soporte real. Programación mediante CONNECT I/O, el cual mantiene una comunicación en tiempo real con HOME I/O.
- Desarrollo del software de operación mediante botonera o panel encargado de controlar el simulador HOME I/O. Será programado mediante pasarela en Arduino.

2 DESCRIPCIÓN DEL SOFWARE HOME I/O

Home I/O es un software educativo que simula una casa inteligente y te permite aprender y mejorar sus habilidades en domótica, transferencia de calor, eficiencia energética y mucho más. Puede ser tan simple como controlar cualquiera de los dispositivos disponibles, como una puerta de garaje motorizada o un termostato, o tan sofisticado como configurar una red para permitir el control centralizado de toda la casa.

Fue creado por la empresa portuguesa Real Games, desarrollador de software educativo 3D para escuelas, universidades y otras organizaciones. Real Games ha estado trabajando en software de simulación durante diez años estableciendo sólidas conexiones de investigación y desarrollo con universidades y centros de investigación.

2.1 HOME I/O

Diseñado para cubrir un amplio rango de objetivos curriculares dentro de la Ciencia, Tecnología, ingeniería y matemáticas (STEM, por sus siglas en inglés), el simulador HOME I/O tiene todo lo que necesitas crear y un monitor en tiempo real inteligente de la simulación de la casa domótica.

Figura 2-1 Emblema del software HOME I/O

Con una consola doméstica incorporada, es posble monitorizar, controlar y asegurar una casa virtual a partir de la creación de inteligentes escenarios en la casa. Alternativamente, se permite usar CONNECT I/O e integrar HOME I/O con un amplio rango de tecnologías de automatización externas: PLC, SoftPLC, Modbus, OPC, microcontroladores, arduino, etc.

Figura 2-2 Vistas exteriores de la casa inteligente

- **Consola inteligente**: Crear escenarios inteligentes en la casa usando comunes dispositivos de automatización. Como, por ejemplo, ajustar luces para apagar o encender cuando la gente entra o abandona una habitación o chequear si las puertas y ventanas están abiertas o cerradas.

- **Parámetros dinámicos**: Cambiar los ajustes del tiempo, día, hora y localización. Mira como estos parámetros afectar durante la simulación y analiza los resultados.

- **Más rápido que en tiempo real**: El sistema nos permite acelerar el tiempo y simular un día en 17 segundos.

- **174 dispositivos interactivos**: HOME I/O permite interactuar con luces, calefactores y otros inteligentes dispositivos de la casa usando más de 400 puntos de entradas/salidas.

- **Comportamiento térmico**: Simulación a tiempo real del comportamiento térmico introduciendo condiciones diferentes de tiempo, ubicación y propiedades del edificio.

- **Entradas y salidas digitales y analógicas**: Encender y apagar luces usando valores digitales o medidas de brillo con valores analógicos.

- **Ambiente de alrededor:** Cambiar condiciones del tiempo, ubicación, día, hora y otros parámetros.

- **Consumo de energía**: El panel datos de energía en tiempo real acerca del consumo de energía y contador en función de la hora, día, semana y mes acerca del consumo de energía y coste.

- **Conectar tecnologías**: CONNECT I/O permite integrar HOME I/O con una amplia variedad de tecnologías de automatización externas (Por ejemplo, PLC, Modbus, microcontroladores y otros).

2.2 CONNECT I/O

CONNECT I/O es un software libre del tipo SoftPLC que es capaz de comunicarse con diferentes tipos de tecnologías. Para nuestro proyecto CONNECT I/O será una herramienta que permitirá a HOME I/O ser fácilmente integrada con tecnologías de automatización. Con CONNECT I/O tu implementas algunos tipos de funcionalidades por dibujos de diagramas de bloques o nodos enlazándolos juntos entre ellos.

CONNECT I/O abarcará los siguientes tipos de programación:

- Funciones de aritmética básica
- Funciones de puertas lógicas
- Funciones Booleanas
- Empleo de funciones Set & Reset: SR & RS
- Bloques de comparación
- Temporizadores TON Y TOFF
- Contadores CTU, CTD & CTUD
- Codificadores y descodificadores: Bit a Byte y viceversa
- Triggers de subida y bajada
- Bloques DAQ conexión entre CONNECT I/O y PLC

CONNECT I/O permitirá la creación y control del flujo de datos entre puntos de entrada y salida. Esta aplicación nos permite ser conectada entre software y hardware logrando trabajar como puente entre ambos.

Básicamente, se usa CONNECT I/O para tres diferentes propósitos:

- Conectar HOME I/O a tecnologías de automatización externas (Por ejemplo PLC, Modbus, microcontroladores, etc) En este caso, CONNECT I/O puede ser visto como una puerta entre la tecnología externa y la simulación.
- Controlar HOME I/O por diseño de un controlador por funciones de bloques.
- Realizar adquisición y análisis de datos provenientes de HOME I/O.

Por supuesto, estas son las utilidades más comunes donde podemos emplear el software. CONNECT I/O es increíblemente flexible, pudiendo usar todas estas funcionalidades a la misma vez e incluso permite extenderlas con tu propio código desarrollado mediante plugins.

2.3 ENGINE I/O

ENGINE I/O es el sistema que permite el intercambio de datos con aplicaciones externas, es decir, es el encargado de la comunicación en tiempo real entre la interfaz CONNECT I/O y otros softwares desarrollados por Real Games. En nuestro caso, dicho programa será HOME I/O. Siendo este el motivo principal de su gran consumo de recursos.

La aproximación de Real Games a las simulaciones permite desarrollar un entorno que pueda ser accesible y expandido por cualquiera. De ahí su gran uso por estudiantes en institutos e ingenieros en la universidad.

El motor ENGINE I/O es un montaje .NET Framework 2.0, el cual nos permite interprocesar comunicación (IPC) entre simulaciones y tecnologías externas como pueden ser software (Matlab, LabVIEW, etc.) o hardware (Arduino, PICAXE, etc.).

2.4 Programación mediante CONNECT I/O

Si eres nuevo en CONNECT I/O, será muy útil presentar algunas de sus características básicas de programación y organización previamente a introducirnos de lleno en la automatización de los distintos elementos objetos de este proyecto.

2.4.1 Introducción básica a la programación en CONNECT I/O

Comenzaremos analizando la interfaz de usuario del programa CONNECT I/O. Está formada por:

1. Menú. Ofrece un acceso general a los comandos.
2. Herramientas. Controlan la ejecución del programa. Se usan para empezar, pausar, resetear o realizar un ciclo de actualización de diagrama único.
3. Panel de Nodos. Lista jerárquica de todos los nodos agrupados por tipos de nodos
4. Panel de programación. En esta sección será donde dibujemos nuestros distintos diagramas.
5. Perspectiva general. Ofrece una representación visual de todo el diagrama.
6. Propiedades. Muestra las propiedades editables y documentación del nodo seleccionado.
7. Registros. Muestra información útil acerca del elemento seleccionado.

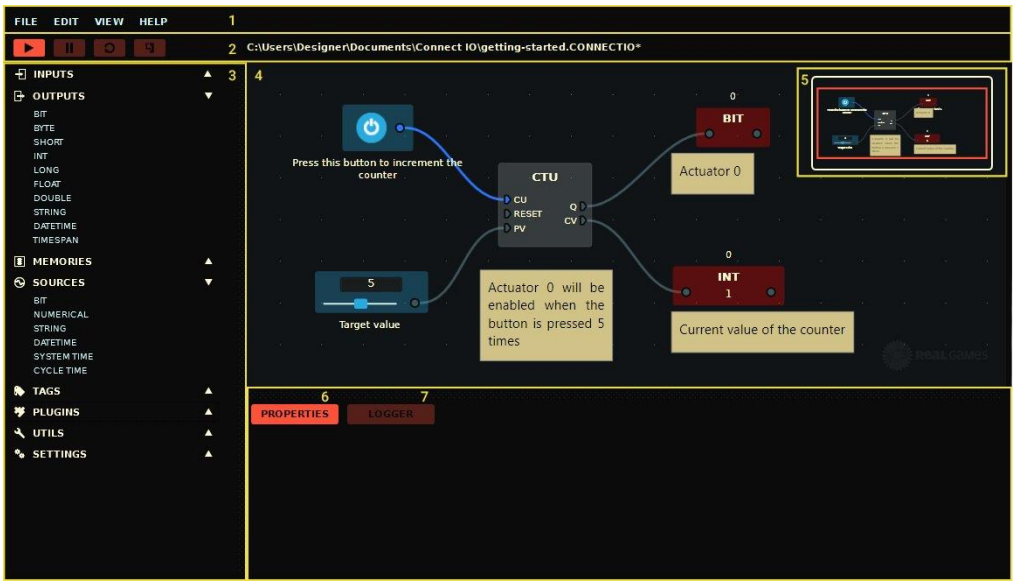


Figura 2-3 Interfaz del software CONNECT I/O

2.4.2 Programación basada en nodos

Todos los nodos están hechos de sockets que son usados para unir distintos nodos juntos. Salidas de sockets puede únicamente estar unido a una entrada de socket, no es posible crear bucles mediante unión de sockets de salida a sockets de entrada directamente. Para crear bucles en tus diagramas, tendrás que usar memorias.

Tabla 2-1 Clasificación de los distintos tipos de nodos en CONNECT I/O

Tipo de Nodo	Descripción
Entradas (Input)	Usados para leer valores de los sensores de HOME I/O. Por ejemplo el estado de un interruptor en función de si encuentra apagado o encendido.
Salida (Output)	Usados para escribir valores de actuadores en HOME I/O. Por ejemplo encender o apagar una luz.
Memoria (Memories)	Usados para almacenar valores durante la ejecución de un diagrama o, en algunos casos especiales, leer o escribir valores de/en simulaciones.
Fuente (Sources)	Usados como fuente de valores, encontrarás un nodo diferente por cada tipo de dato soportado.
Etiqueta (Tags)	Nodos correspondientes a cada uno de los sensores y actuadores que aparecen en la simulación. Leen y escriben valores intercambiados con la simulación.
Plugin	Nodos que proporcionan algún tipo de funcionalidad. Por ejemplo operaciones aritméticas, contadores, conectividad con PLC y otros similares. Por ejemplo conexión con un PLC Siemens a través de Ethernet o Arduino como es el objetivo del proyecto.

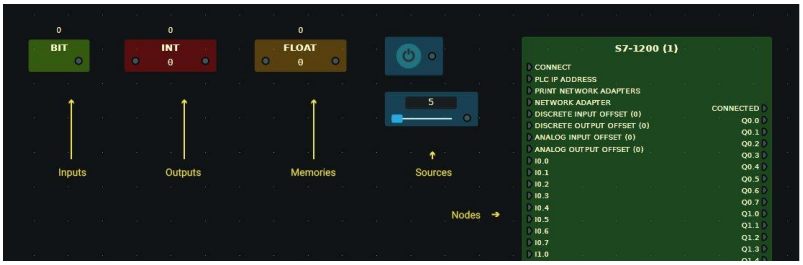


Figura 2-4 Simbología de los distintos tipos de nodos

Cada uno de los datos, ya sea una entrada, salida o memoria (Por ejemplo: Bit, Byte, Short, Int, Long, etc.) tiene asociada una dirección de memoria.

Por tanto, para usar diferentes entradas de tipo Bit, tendremos que asociarlas a distintas direcciones. Por ejemplo una de ellas a la dirección 0 y la otra a la 1. Esta sería la forma para trabajar con las sucesivas entradas tipo Bit o cualquier otra de diferente tipo.

2.4.3 Cómo activar nodos asociados a sensores y actuadores en HOME I/O

A medida que nos desplacemos por la casa simulada por HOME I/O encontramos multitud de actuadores (Interruptores con distintas funciones, puertas, persianas, controladores de la temperatura) y sensores (movimiento, posición, topes en el movimiento de la puerta del garaje o reja externa, etc). Todos ellos tendrán asociado un color: rojo, verde o azul. En función de dicho color podremos conocer a qué estado o tipo de programación están asociados; los cuales son:

- Actuadores o sensores manuales accionándolos a partir del simulador HOME I/O.

Es decir, a medida que nos desplacemos por las distintas habitaciones de la casa podremos ir activando los distintos actuadores virtuales, mientras los sensores por lo general su activación no generará ningún tipo de acción. Estos elementos estarán marcados en color rojo. Al comenzar una nueva partida en HOME I/O todos los elementos vendrán en este estado por defecto.

Figura 2-5 Modo manual

- Activado para programación automática mediante Tablet Virtual. Estos elementos vendrán marcados en color verde y al ser activados nos aparecerán en el listado de elementos listos para ser programados mediante este soporte.

Figura 2-6 Modo de programación Tablet Virtual

Una vez cambiado el estado a verde, los dispositivos activados nos aparecerán ordenados por pestañas según cuál sea su función, y dentro de cada una de las pestañas, listados y ordenados por habitaciones.

Estas pestañas se sitúan la parte superior de la interfaz de la Tablet Virtual de HOME I/O. Según la función del elemento encontramos las siguientes secciones: Iluminación, calefacción, seguridad, etc.

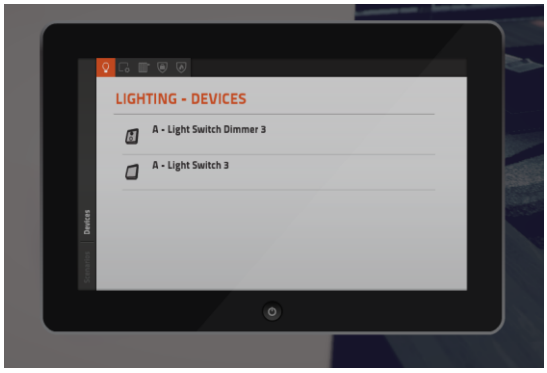


Figura 2-7 Diferentes secciones de dispositivos

Asimismo, en el lateral izquierdo de la interfaz de la Tablet encontramos otras dos pestañas: La superior llamada “Devices” descrita anteriormente y ya puesta por defecto, y “Scenarios”.

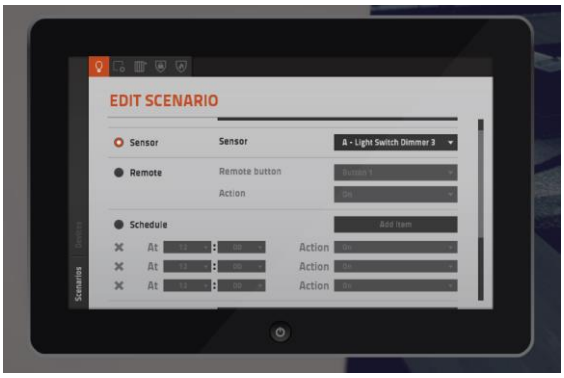


Figura 2-8 Seleccionando sensor en escenario

En esta sección podemos programar las acciones que queremos que se lleven a cabo por cada uno de los elementos previamente seleccionados.

"Scenarios" nos ofrece infinidad de posibilidades de programación sin necesidad de usar el software CONNECT I/O: desde usar el mando inalámbrico a programar una agenda de acciones para que se lleven a cabo en algún momento del día.

Puesto que no es el objetivo del trabajo, no se ha profundizado mucho en este tipo de programación.

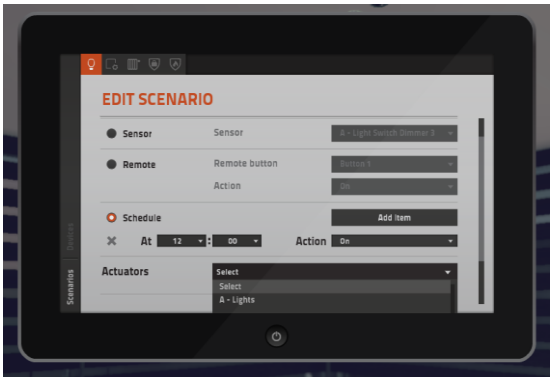


Figura 2-9 Programar escenario

- Elementos activados para programación mediante CONNECT I/O. Es el estado en el que se enfocará principalmente el proyecto. El color asociado a este estado es el azul. Una vez activados los dispositivos en HOME I/O nos aparecerán en el listado de etiquetas (Tags) de CONNECT I/O permitiéndonos jugar con dichos sensores o actuadores para que tenga lugar un determinado efecto.

Figura 2-10 Modo programable en CONNECT I/O

Desplegando en la interfaz de CONNECT I/O la sección de etiquetas (Tags) podremos encontrar los distintos elementos activados para programar mediante bloques.

Cada elemento irá precedido de una letra que nos identificará dónde y en qué parte de la casa se

encuentran. Esto será de gran ayuda ya nos permitirá diferenciar por ejemplo los nombres de los sensores de movimiento de distintas habitaciones.

A continuación podemos ver una captura de ejemplo del listado de etiquetas y del mapa de la casa con la letra identificadora que corresponde a cada habitación:

Figuras 2-11 Mapa e identificadores de los distintos Tags o nodos

Para modificar el estado de un elemento, tan sólo tenemos que pulsar sobre el color en HOME I/O y automáticamente se verá modificado, siendo el orden el marcado en la descripción anterior: rojo por defecto, verde a continuación, azul en último lugar, y vuelta al origen.

2.4.4 Construcción de diagramas de bloques

Una vez que ya tengamos en el listado de "Tags" los elementos que queremos automatizar en función de la activación de algún sensor o actuador, nos disponemos a construir el diagrama.

Para ello, previamente habremos tenido que analizar con detenimiento cuál es la meta a alcanzar, es decir, si queremos por ejemplo encender la luz de la entrada de forma automática, debemos preguntarnos: ¿cómo pretendemos conseguirlo? y ¿qué sensores nos serán útiles en nuestro propósito?

Una vez realizado este primer análisis y colocado dichos sensores y/o actuadores en el estado de programación para CONNECT I/O debemos conocer cómo funcionan sus nodos. Por ejemplo, el sensor de movimiento ("Motion Detector") una vez activado su señal se mantiene únicamente en HIGH durante 5 segundos. Para ello será necesario un bloque RS que mantenga dicha señal en HIGH hasta que el RESET sea activado de alguna manera.

Por otro lado, también debemos analizar su funcionamiento mecánicamente. El caso mas significativo es el de los interruptores. Existen dos tipos: actuantes y pulsadores. La diferencia fundamental reside en que los primeros están siempre abiertos o cerrados, mientras los segundos son sólo momentáneos.

2.4.5 Cómo hacer bucles

Como tampoco queremos que la luz quede encendida de manera indefinida, necesitamos encontrar una condición que active la señal RESET y corte la alimentación del bloque actuador, en nuestro ejemplo el nodo de la luz de la entrada ("Lights Entrance").

La solución más sencilla es implementar un temporizador TON que nos marque la caída de la señal una vez cumplido ese tiempo. Cuando se cumpla, la salida Q realizará una realimentación al bloque RS mediante un bit auxiliar y pondrá a LOW el bit que genera la acción de encender la luz.

Por supuesto a medida que vayamos haciendo modificaciones, debemos ir probando para ver cómo responde el circuito elaborado y si finalmente obtenemos el resultado deseado.

Esta sería la estrategia basada en el método prueba y error que vamos a seguir para programar los distintos elementos tecnológicos que constituirán nuestro hogar domótico.

El circuito descrito anteriormente a modo de ejemplo sería el siguiente:

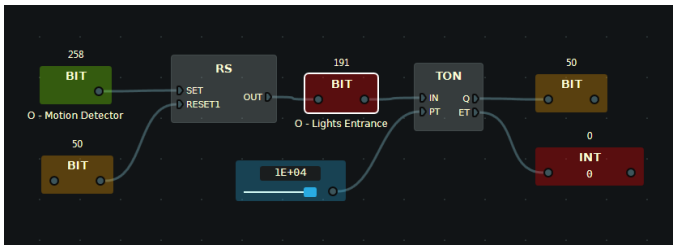


Figura 2-12 Diagrama de ejemplo

3 COMUNICACIÓN ENTRE ARDUINO Y CONNECT I/O

En este capítulo vamos a explicar cómo utilizar un Arduino Uno como bloque USB de entradas/ salidas digitales y analógicas. Se hace posible así disponer de un sistema de adquisión de datos DAQ USB E/S realmente barato. Para ello tendremos que desarrollar 2 plugins Arduino para poderlos usar durante la programación en el software CONNECT I/O, los cuales son:

- DAQUINO_DIGIT E/S. Plugin con entradas y salidas digitales basadas en el sistema todo ó nada.

- DAQUINO_DIGIT_ANALOG. Plugin con entradas analógicas y salidas todo o nada.

Estos plugins permiten la comunicación con Arduino Uno a través de la conexión USB y el intercambio de los valores de E/S. Esto nos hace posible llevar a cabo la interacción entre el mundo real y el mundo virtual generado por HOME I/O.

3.1 Descripción del hardware empleado

3.1.1 Arduino

Arduino es una plataforma de hardware libre, basada en una placa con un microcontrolador y un entorno de desarrollo, diseñada para facilitar el uso de la electrónica en proyectos multidisciplinares.

Figura 3-1 Emblema del software Arduino

Por otro lado Arduino nos proporciona un software consistente en un entorno de desarrollo (IDE) que implementa el lenguaje de programación de arduino y el bootloader ejecutado en la placa. La principal característica del software de programación y del lenguaje de programación es su sencillez y facilidad de uso.

¿Para qué sirve Arduino? Arduino se puede utilizar para desarrollar elementos autónomos, conectándose a dispositivos e interactuar tanto con el hardware como con el software. Nos sirve tanto para controlar un elemento, pongamos por ejemplo un motor que nos suba o baje una persiana basada en la luz existente es una habitación, gracias a un sensor de luz conectado al Arduino, o bien para leer la información de una fuente, como puede ser un teclado, y convertir la información en una acción como puede ser encender una luz y pasar por un display lo tecleado.

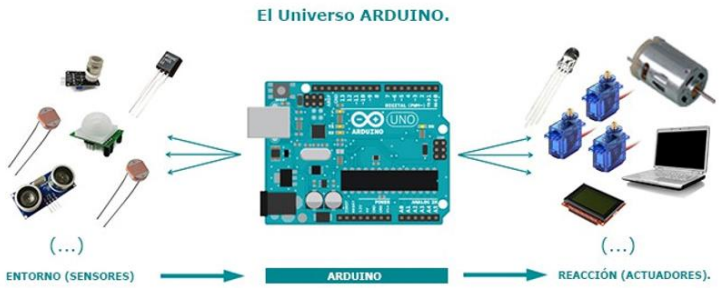


Figura 3-2 Posibilidades ofrecidas por Arduino

Hay otro factor importante en el éxito de Arduino, es la comunidad que apoya todo este desarrollo, comparte conocimiento, elabora librerías para facilitar el uso de Arduino y publica sus proyectos para que puedan ser replicados, mejorados o ser base para otro proyecto relacionado.

3.1.1.1 Hardware

El hardware de Arduino es básicamente una placa con un microcontrolador. Un microcontrolador (abreviado µC, UC o MCU) es un circuito integrado programable, capaz de ejecutar las órdenes grabadas en su memoria. Está compuesto de varios bloques funcionales, los cuales cumplen una tarea específica. Un microcontrolador incluye en su interior las tres principales unidades funcionales de una computadora: unidad central de procesamiento, memoria y periféricos de entrada/salida.

Características de un Microcontrolador:

- Velocidad del reloj u oscilador
- Tamaño de palabra
- Memoria: SRAM, Flash, EEPROM, ROM, etc.
- I/O Digitales
- Entradas Analógicas
- Salidas analógicas (PWM)
- DAC (Digital to Analog Converter)
- ADC (Analog to Digital Converter)
- Buses
- UART
- Otras comunicaciones.

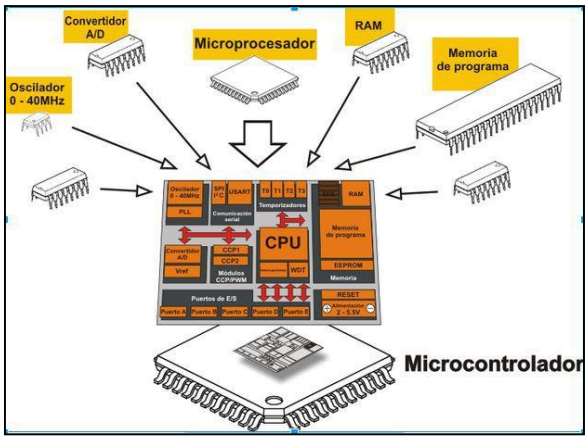


Figura 3-3 Características de un microcontrolador

Arduino dispone de una amplia variedad de placas y shields para usar dependiendo de nuestras necesidades. Para la realización de nuestro proyecto hemos escogido la placa Arduino UNO:

Figura 3-4 Placa Arduino

Un shield es una placa compatible que se puede colocar en la parte superior de los arduinos y permite extender las capacidades del arduino.

Las shields se pueden comunicar con el arduino bien por algunos de los pines digitales o analógicos, o bien por algún bus como el SPI, I2C o puerto serie, así como usar algunos pines como interrupción. Además, estas shields se alimentan generalmente a través del Arduino mediante los pines de 5V y GND.

Cada Shield de Arduino debe tener el mismo factor de forma que el estándar de Arduino con un espaciado de pines concreto para que solo haya una forma posible de encajarlo.

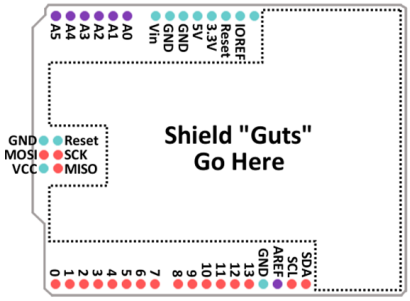


Figura 3-5 Esquema de la placa Arduino y la posición del Shield

3.1.1.2 Software

El software de Arduino es un IDE**,** entorno de desarrollo integrado (siglas en inglés de Integrated Development Environment). Es un programa informático compuesto por un conjunto de herramientas de programación.

El IDE de Arduino es un entorno de programación que ha sido empaquetado como un programa de aplicación; es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI). Ademas incorpora las herramientas para cargar el programa ya compilado en la memoria flash del hardware.

Figura 3-6 Interfaz del software Arduino

Aunque se hable de que hay un lenguaje propio de programación de Arduino, no es cierto, la programación se hace en C++ pero Arduino ofrece unas librerías o core que facilitan la programación de los pines de entrada y salida y de los puertos de comunicación, así como otras librerías para operaciones específicas. El propio IDE ya incluye estas librerías de forma automática y no es necesario declararlas expresamente.

Además del IDE estándar existen una buena variedad de programa de IDE alternativos para trabajar con Arduino. Su instalación y uso no es tan sencillo como el anterior, pero a cambio tendremos funcionalidades adicionales, mayor potencia, y mejor control del código ejecutado. Algunos de los entornos alternativos más empleados son: Visual Studio, Eclipse, Atmel Studio 6.0, etc.

3.1.1.3 Fritzing

Fritzing es el programa por excelencia para la realización de esquemas eléctricos en proyectos con Arduino. Dispone bibliotecas con la mayoría de componentes, incluido por supuesto los propios Arduino, placas de conexiones, led, motores, displays etc. También podemos importar piezas hechas por otras personas de la comunidad o crear las nuestras propias. Sin duda alguna, una herramienta muy útil en el desarrollo de nuestro proyecto.

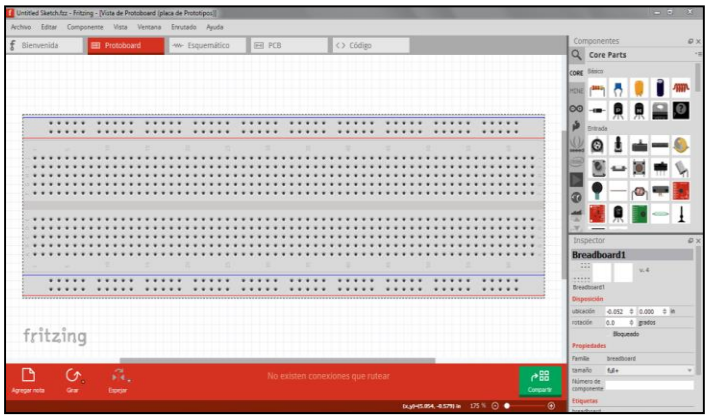


Figura 3-7 Interfaz del software Fritzing

3.1.2 Potenciómetro

El potenciómetro nos proporciona una resistencia variable según vayamos modificando su posición. Si está totalmente cerrado obtendremos como salida el máximo voltaje (el de entrada), si lo tenemos totalmente

abierto, obtendremos 0 voltios y si lo tenemos en una posición intermedia obtendremos una fracción del voltaje de entrada proporcional a la posición en la que se encuentre. Este comportamiento se llama divisor de tensión.

Existen dos tipos de potenciómetros: lineal y logarítmicos. Nosotros vamos a utilizar uno lineal, es decir, la resistencia es directamente proporcional al ángulo de giro.

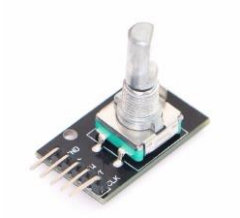


Figura 3-8 Potenciómetro lineal

Este tipo de potenciómetro dispone de 3 patillas. Según el potenciómetro que utilicemos, deberemos identificar la funcionalidad de cada patilla. Una patilla irá conectada a la fuente de alimentación, otra a tierra o GND y por último la tercera patilla será la salida del potenciómetro.

Modo de Conexión:

1. Alimentación de 5V.
2. Señal de salida al pin analógico del Arduino A0, A1 ó A2.
3. GND

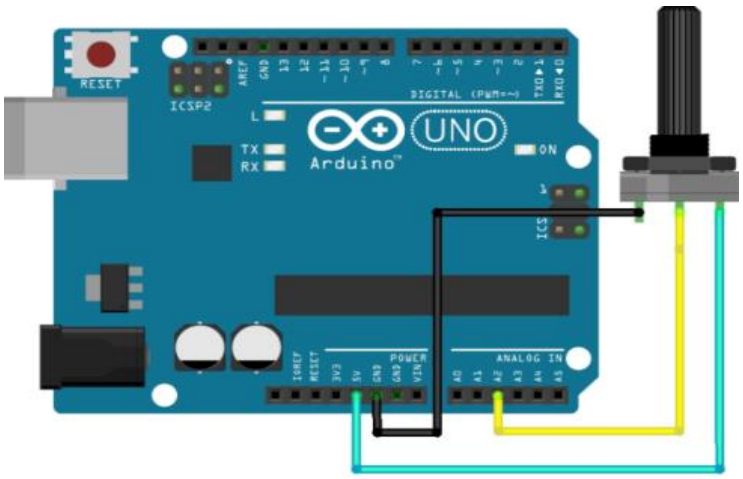


Figura 3-9 Esquema de la conexión del potenciómetro

3.1.3 Módulo Joystick

Funciona igual que el joystick del mando de una videoconsola. Permite controlar las direcciones X, Y y Z. Está considerada como la combinación de potenciómetros y de un botón, normalmente abierto. Los tipos de datos de las direcciones X e Y son señales de entradas analógicas y la dirección Z es una entrada de señal digital. Así los puertos X e Y se conectan a los pines analógicos del Sensor Shield, mientras que el puerto Z se conecta al pin digital.

Figura 3-10 Módulo joystick

Modo de conexión:

1. GND
2. Alimentación de +5 Vcc
3. Control en la dirección X: vRX
4. Dirección Y: vRY
5. Botón SW

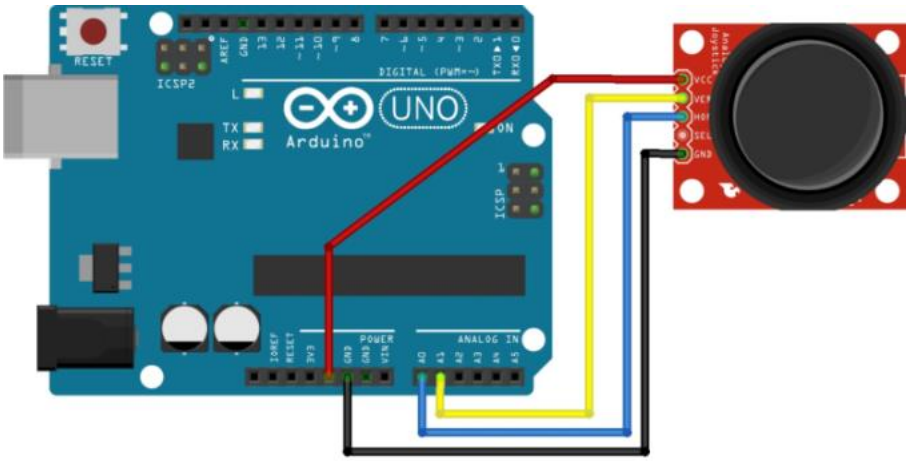


Figura 3-11 Esquema de la conexión del joystick a la placa Arduino

3.1.4 Pulsadores y Resistencias

- Interruptores con forma de botón x5. Dispone de cuatro terminales.

  Los terminales del pulsador están conectados 2 a 2: De forma que un terminal de un lateral este con otro del lateral contrario. Si el botón está pulsado, los 4 están conectados entre sí.

Figura 3-12 Interruptor pulsante

- Resistencias x5.
  Resistencias x5 con valor de 1 KΩ y de Tolerancia ± 5%.

Figura 3-13 Resistencias de valor 1 KΩ

En función de cómo coloquemos las resistencias estableceremos un estado lógico en un pin o entrada de un circuito lógico cuando se encuentra en estado reposo. Como bien indica su nombre la resistencia

pull up establece un estado HIGH y las resistencias pull down establecen un estado LOW cuando el pin se encuentra en reposo.

En nuestro caso optaremos por un montaje pull down, es decir el Pin digital al que conectemos el pulsador botón esta unido permanentemente a GND y recibirá 5 voltios al pulsar el botón.

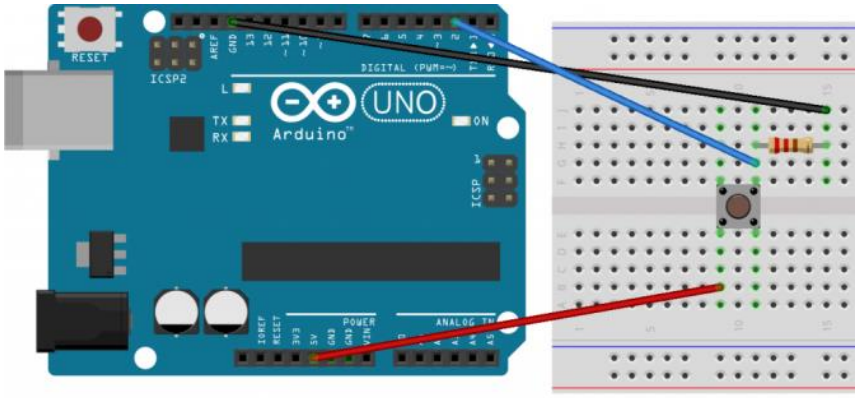


Figura 3-14 Montaje de un botón conectado en pull-down a la placa Arduino

3.2 Funcionalidad

El plugin DAQUino permite activar los pines de salida basados en todo o nada del Arduino y de recuperar el valor de las entradas del mismo. El abanico de posibilidades ofrecidas son muchísimas para HOME I/O. Podemos, por ejemplo, desde utilizar los sensores reales con un comando llevado a cabo en CONNECT I/O, hasta alternativamente utilizar los distintos actuadores virtuales que encontraremos por la casa, o conectarnos a una parte de control externa (microcontrolador, autómata programable industrial, etc) el cual programaremos con una misión determinada a llevar a cabo.

Las entradas físicas de la placa Arduino son, por tanto, salidas del complemento. De la misma forma, las entradas del plugin son las salidas físicas de la placa Arduino.

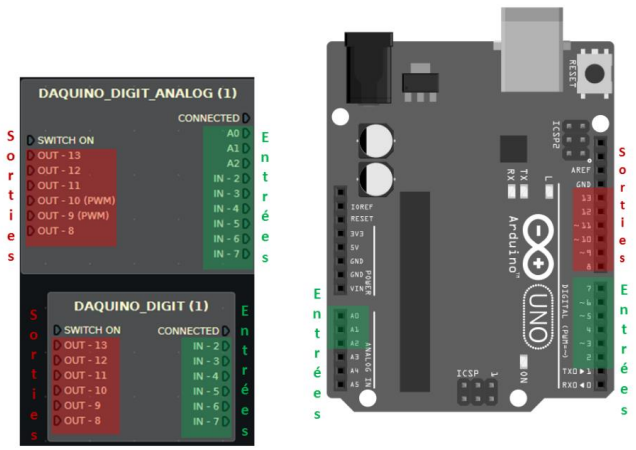


Figura 3-15 Relación entre las entradas y salidas del nodo y de la placa

Gracias a estos complementos, será posible interconectar las partes operativas simuladas de la aplicación HOME I/O con:

- Los sensores o actuadores reales que irán conectados a la placa Arduino (LEDs, botones, interruptores, sondas de temperatura, etc)
- Controladores externos para externalizar la parte de control vinculada a un sistema de automatización como los mencionados anteriormente.

El programa que se cargará en el Arduino es un «Parser» (Analizador de sintaxis) que permite comunicarse con el PC a través de la conexión USB. De esta forma, es responsable de leer la información presente en sus puertos de entrada y la envía a CONNECT I/O. De la misma manera, escribe en sus puertos de salida la información enviada por CONNECT I/O. El intercambio de información entre el Arduino y CONNECT I/O se realiza por la conexión serie USB vía el puerto COM256 que debemos configurar previamente.

3.3 Instalación de plugins DAQuino

El modulo de E/S a estudiar en este documento se utiliza en el software CONNECT I/O como complemento. Corresponde a un archivo de biblioteca de clases (.dll) que inicializa atributos e implementa métodos para comunicar información a través de la conexión serie con la interfaz de E/S de Arduino. Desde el punto de vista del complemento –DAQUino- en CONNECT I/O, la información presente en la salida del bloque representa los estados actuales de las variables de tiempo real relacionadas con HOME I/O mientras que la información de entrada representa las cantidades asociadas con los dispositivos hardware (sensores, actuadores) con la simulación a través del módulo de E / S.

Para disponer del plugin Arduino en CONNECT I/O, los dos dll (bibliotecas de clases) <DAQUino*> (Datos de adquisición de Arduino) siguientes son necesarios:

- DAQUinoDigit.dll : Representa la biblioteca de clases necesaria en CONNECT I/O para utilizar el módulo E/S de tipo digital dentro de su área de programación gráfica.

- DAQUinoDigitAnalog.dll : Representa la biblioteca de clases necesaria en CONNECT I/O para utilizar el módulo E/S de tipo digital y analógico dentro del área de programación gráfica.

Posteriormente tenemos que colocar estas bibliotecas en el directorio <plugin> de la carpeta de instalación CONNECT I/O, para que los plugins estén activos dentro de CONNECT I/O. Para ello, arrastre y suelte estos archivos en el directorio "DAQUino" del directorio pluggins de la carpeta de instalación de CONNECT I/O. Por ejemplo "*C:\Program Files (x86)\Real Games\Connect IO\Plugins\DAQuino*".

3.4 Ajustar Puerto COM256 del periférico

Para que la interfaz de comunicación entre los dos componentes, el puerto COM 256 del PC y el dispositivo Arduino UNO, sea más robusta, es preferible fijar el puerto COM del PC asociado con este dispositivo a un índice fijo constante y conocido por los complementos DAQUINO. Dado que el índice de puerto COM está codificado en 8 bits, usaremos el índice máximo de 256 para superar problemas de puertos existentes. Para ello, debemos haber instalado previamente los controladores Arduino encapsulados con el IDE. Además, debe tener derechos de administrador para realizar esta configuración.

- Conecte la tarjeta Arduino UNO al PC.

- Ir a Panel de control \ Hardware y audio \ Periféricos e impresoras

- Haga clic derecho en el "Arduino Uno" dispositivo, a continuación, haga clic en "Propiedad"

- Haga clic en la pestaña "hardware" y luego haga clic en el botón "propiedad".

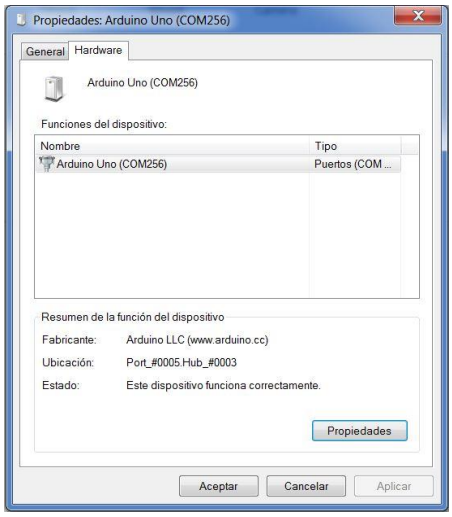


Figura 3-16 Propiedades COM256

- En la pestaña "Puerto parámetro" haga clic en "Avanzado ..."

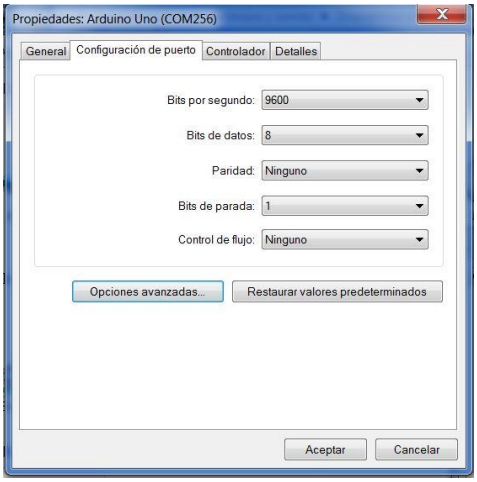


Figura 3-17 Opciones avanzadas COM256

- Asigne el número de puerto a COM256 en la lista desplegable "Número de puerto COM ".

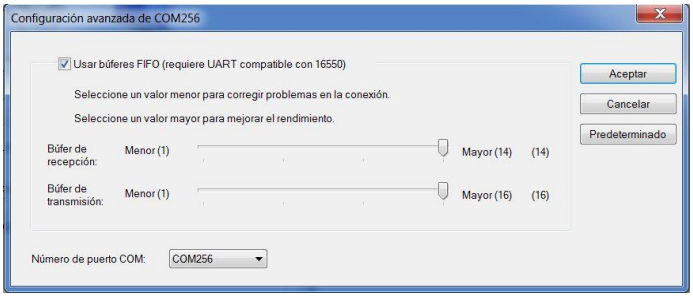


Figura 3-18 Número de puerto

- Por último, sabremos si hemos realizado correctamente el procedimiento si en "periféricos e impresoras" nos aparece el dispositivo asociado a la placa con el puerto COM256.

Figura 3-19 Dispositivo en panel de control

Nótese que todas estas capturas están hechas en el sistema operativo Windows 7.

3.5 Arquitectura del sistema

La arquitectura material obedece al principio del modelo maestro-esclavo e integra de esta forma a dos sistemas informáticos. El protocolo de comunicación entre estas 2 entidades utiliza el canal de conexión serie USB COM256.

- El ordenador juega un papel de maestro y envía los comandos y las consultas al esclavo que éste ejecuta.
- El microcontrolador funciona como esclavo, recibe los comandos y las consultas del maestro, los ejecuta y devuelve el resultado (en el caso de una consulta) a través del mismo canal que recibió el comando: el enlace serie USB a través del puerto COM 256.

3.6 Motor de entrada/ salida

Ademas gracias a la tecnología ENGINE I/O, una cierta parte de la memoria atribuida a la ejecución de las aplicaciones es compartida y común entre sí. Este principio de funcionamiento permite a estas dos aplicaciones intercambiar la información.

Plugin DAQUino.dll

El modulo de E/S a estudiar en este documento se utiliza en el software CONNECT I/O como complemento. Corresponde a un archivo de biblioteca de clases (.dll) que inicializa atributos e implementa métodos para comunicar información a través de la conexión serie con la interfaz de E/S de Arduino. Desde el punto de vista del complemento –DAQUino- en CONNECT I/O, la información presente en la salida del bloque representa los estados actuales de las variables de tiempo real relacionadas con HOME I/O mientras que la información de entrada representa las cantidades asociadas con los dispositivos hardware (sensores, actuadores) con la simulación a través del módulo de E / S.

Esquema del montaje y funcionamiento se presenta en la siguiente figura.

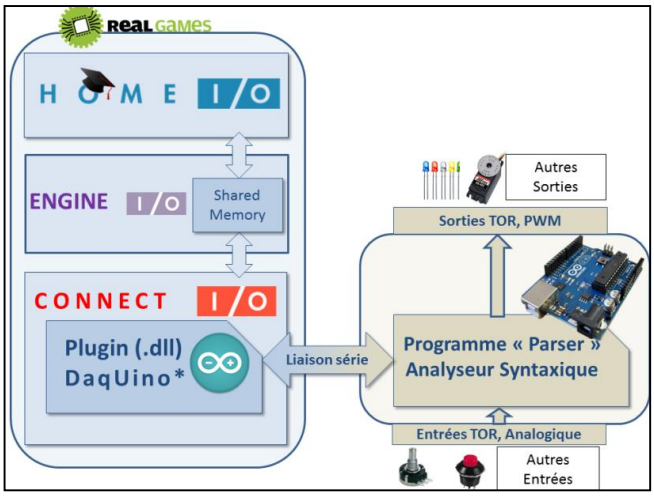


Figura 3-20 Esquema global de funcionamiento y elementos conectados

4 APLICACIÓN MEDIANTE DEMOSTRACIÓN DESARROLLADA: ELEMENTOS ANALÓGICOS

"Talk is cheap. Show me the code."

- Linus Torvalds-

4.1 Luces mediante potenciómetro

Para la configuración de un potenciómetro conectado a arduino que simule el dimmer de nuestra casa inteligente, en primer lugar, tenemos que cambiar el estado de los interruptores a programable en CONNECT I/O como se explico en el punto anterior. En nuestro caso, hemos decidido trabajar con los interruptores del salón:

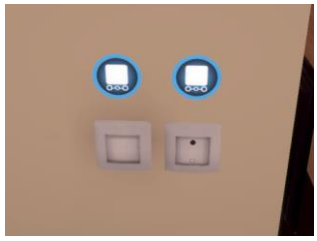


Figura 4-1 Activación del nodo de los interruptores

I/O Señales:

- Entradas:
  - Bloque DAQUINO_DIGIT_ANALOG
- Salidas:
  - Bloque medidor de la salida. Float 132.
  - Luces dimmer. Float 0.
- Otras funciones:
  - Fuente tipo bit que alimenta al bloque Arduino.
  - Bloque multiplicador.
  - Bloque numérico para introducir la constante.

Explicación de la solución

Una vez ejecutado el paso anterior procedemos a construir el diagrama de bloques. Como se aprecia en la

imagen siguiente, utilizamos el bloque DAQUINO_DIGIT_ANALOG alimentado por una fuente de tipo bit conectada al pin "Switch on".

La salida del pin analógico A0 (que realmente es una entrada de la placa) nos da valores entre 0 y 255 en función del giro del potenciómetro, siendo tipo Float. Por este motivo, como el bloque que nos acciona la luz sólo admite valores entre 0 y 10, necesitamos multiplicar por una constante arbitraria, en nuestro caso 0.04, para conseguir disminuir la salida y que nos permita graduarla linealmente.

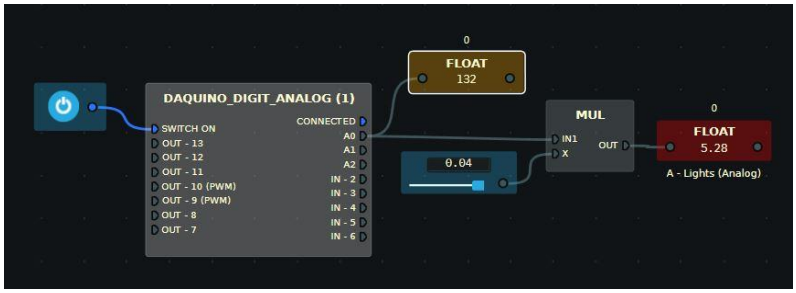
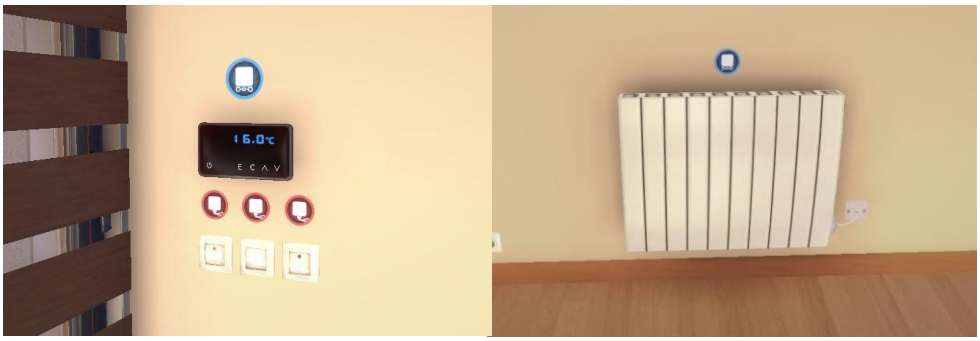


Figura 4-2 Diagrama resultante para la programación de las luces dimmer

4.2 Control de la temperatura de referencia de una habitación mediante potenciómetro

Para controlar la temperatura de la calefacción de cualquier habitación de la casa inteligente, en primer lugar, seleccionamos el calentador y su respectivo panel cambiándolos al estado de programación.



Figuras 4-3 Activación de los nodos del calentador y panel de temperatura

I/O Señales:

- Entradas:
  - DAQUINO_DIGIT_ANALOG
  - Termostato (Thermostat). Indica la temperatura actual de la habitación.
- Salidas:
  - Calentador (Heater). Bit 9.
- Otras funciones:
  - Fuente tipo Bit. Para alimentar al bloque Arduino.
  - Bloque Auxiliar Float. Para medir la temperatura de la referencia.
  - Bloque Numérico para introducir una constante.
  - Bloque División.

- Bloque Lógico.

Explicación de la solución

Al igual que en el montaje anterior tenemos que usar el bloque DAQUINO_DIGIT_ANALOG para poder comunicar el potenciómetro con el diagrama. Como los valores que nos da la salida del pin analógico A1 (entrada para el Arduino) oscilan entre 0 y 255, optamos por dividir el valor de la salida entre una constante para disminuirlo considerablemente y así obtener un valor razonable para una referencia de temperatura. En la imagen siguiente podemos observar como dicha referencia tenía un valor de 16.38ºC.

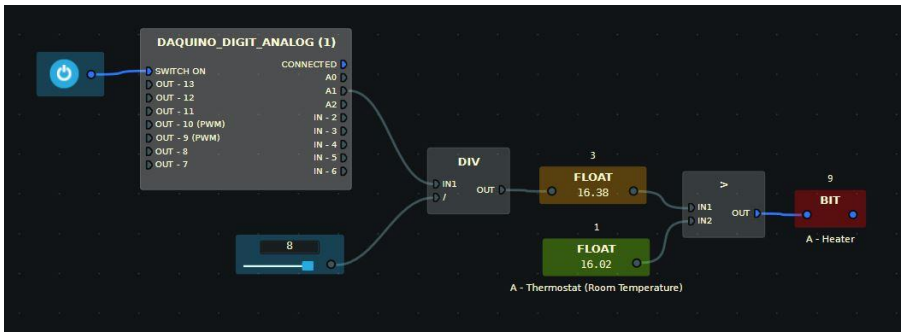


Figura 4-4 Diagrama para la programación del control de la temperaura de una habitación

El sistema se basa en controlar la temperatura de la habitación para que siempre ronde el valor de la referencia. De forma que si la temperatura de la habitación es menor que la referencia marcada por el potenciómetro automáticamente se accionará el calentador hasta alcanzarla. Una vez superado el valor se desactivará. Y así sucesivamente.

4.3 Persianas automáticas mediante joystick

Este sería el último montaje de la parte analógica programada del proyecto. Consiste en el control de las persianas de cualquier habitación mediante un joystick usando uno sus ejes para controlar la subida, la bajada y la parada de las mismas.

Al igual que en las simulaciones anteriores, esta también la hemos llevado a cabo en el salón, lo cual la hace incluso más compleja, ya que es el único lugar de la casa donde podemos encontrar más de una persiana, cuatro para ser exactos. Por este motivo, hemos decidido hacer más realista el montaje habilitando el eje X del joystick para regular un conjunto de dos persianas orientadas hacia norte del salón, mientras que con el eje Y podremos controlar las dos restantes orientadas hacia el oeste con las mismas funcionalidades.

Como es lógico, lo primordial es conseguir controlar una de las persianas, y posteriormente aplicar esa misma metodología a las demás. Es decir, una vez que lo consigamos, tendremos que hacer algunas variaciones para lograr que ese diagrama controle a un conjunto de dos persianas, para luego posteriormente duplicar ese esquema y realizando las modificaciones oportunas conseguir controlar las persianas restantes también de manera conjunta.

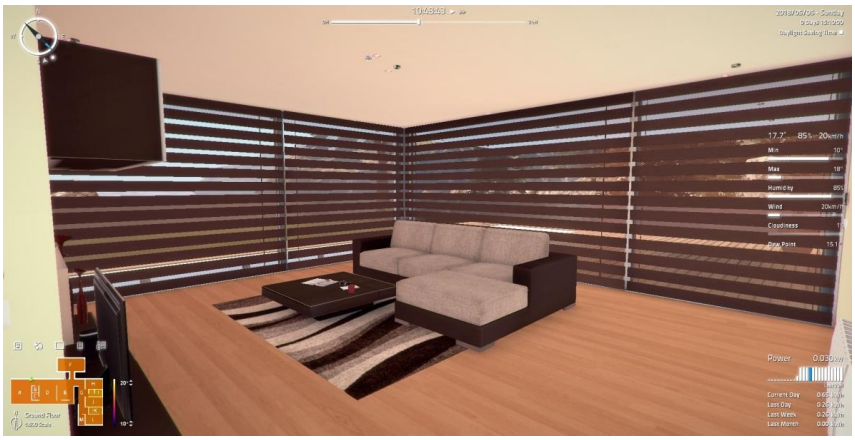


Figura 4-5 Vista de los conjuntos de persianas desde el interior del salón

Siguiendo las mismas pautas que en otros experimentos, en primer lugar tenemos que habilitar como programables los distintos elementos que van a intervenir: cada uno de los sensores de las persianas y sus respectivos interruptores, en el caso del salón dos.

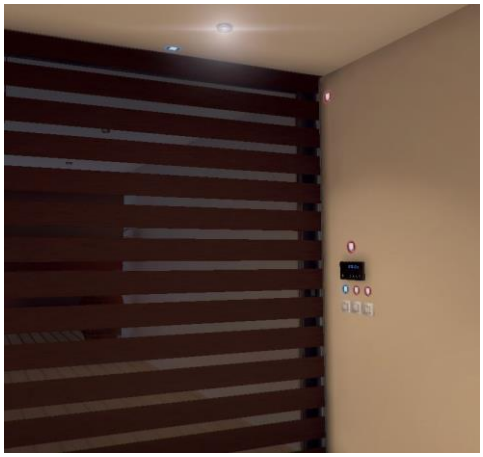


Figura 4-6 Sensores e interruptor necesario para controlar una persiana

I/O Señales:

- **Entradas:**

  - DAQUINO_DIGIT_ANALOG
  - Sensor de posición de la persiana (Roller Shades Openness). Float 3. Valores entre 0 y 10 en función de la altura a la que se encuentre, siendo 10 arriba del todo y 0 bajada completamente.

- **Salidas:**

  - Bloque de acción de subir persianas 1 y 2 (Roller Shade 1 Up y Roller Shade 2 Up)
  - Bloque de acción de bajar persianas 1 y 2 (Roller Shade 1 Down y Roller Shade 2 Down)

- **Otras funciones**:

- Bloques auxiliares a modo de bandera. Bits 2 y 4. Su activación será consecuencia de que se haya accionado previamente la subida/bajada de la persiana.
- Bloque auxiliar a modo de enlace/puente. Se explicará más adelante su utilidad.
- Funciones lógicas: Bloques comparativos (de mayor o menor, igual, etc.) empleados con las variables Float.
- Temporizadores. Su propósito esa dar un leve retraso (0.5 segundos en nuestro ejemplo) para evitar conflictos entre las señales enviadas por el joystick, en primera instancia para accionar la persiana y en segundo lugar para pararlas durante su recorrido.
- Bloques numéricos para introducir constantes.
- Puertas lógicas: NOT, AND, OR.
- Bloque RS para mantener en HIGH de forma permanente la señal que le llega a la salida.

Explicación de la solución

El objetivo consiste en mover el joystick en alguno de los dos sentidos de un eje, por ejemplo hacia arriba sin mantener dicho movimiento en el tiempo, y que la persiana automáticamente suba. Ademas queremos que se detenga durante el trayecto si movemos el joystick en el sentido contrario al anterior, es decir hacia abajo, o en su defecto, porque la persiana haya llegado al final del recorrido, arriba del todo en este caso, y se vea obligada a parar.

Haciendo uso de los bloques descritos anteriormente tenemos que construir un diagrama que satisfaga todas estas condiciones.

En la siguiente imagen podemos ver el esquema de subida de la persiana. Debido a su extensión no hemos incluido el bloque digital-análogico del Arduino para poder centrarnos en la explicación del resto del diagrama.

Como la salida del bloque Arduino nos da una señal Float entre 0 y 255, con valor 0 si movemos el joystick hacia un extremo completamente, y 255 si lo movemos totalmente en el sentido opuesto, para diferenciar ambos movimientos hemos impuesto dos condiciones:

- Si la señal que nos da la salida del Arduino es mayor que el valor impuesto por la constante numérica, 230, estaremos moviendo el joystick hacia arriba, es decir queremos que la persiana suba.
- En cambio, si esta salida es menor que el valor de la otra constante, 50, queremos realizar un movimiento de bajada.

Movimiento de subida

Justo a continuación, para mantener esta señal en HIGH a lo largo del tiempo sin necesidad de estar manteniendo el movimiento del joystick, colocaremos un bloque Set-Reset cuya salida estará conectada directamente a los bloques de acción de subir las persianas de una de las orientaciones (En la imagen persianas 1 y 2) y también a un bloque auxiliar de memoria de tipo Bit que utilizaremos para la realimentación del circuito.

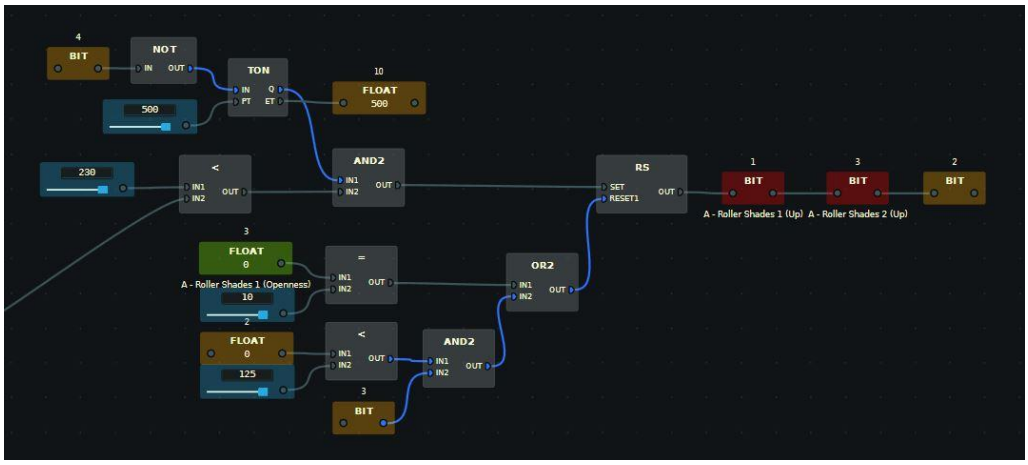


Figura 4-7 Diagrama del movimiento de subida

Como condición para activar el Reset y que pare la alimentación del avance, tenemos que estudiar las siguientes posibilidades:

1) Que hayamos llegado al tope marcado por el sensor de movimiento de la persiana. Como comentamos antes, el tope del sensor en la subida es 10. Para ello emplearemos un bloque lógico con condición de igualdad.
2) Que movamos el joystick hacia abajo con intención de detener el movimiento del avance hacia arriba. Es decir, tenemos que volver a hacer uso del bloque comparador entre una constante arbitraria impuesta por nosotros, 125 en la imagen, y la señal Float del joystick.
   Además, para que esta última condición active el Reset se tiene que cumplir también mediante un AND que el Bit 3 auxiliar de memoria este a 1. Este bloque no es más que un enlace con otra condición que será usada en otros puntos del diagrama, y para abreviarlo se ha implantado de esta manera:

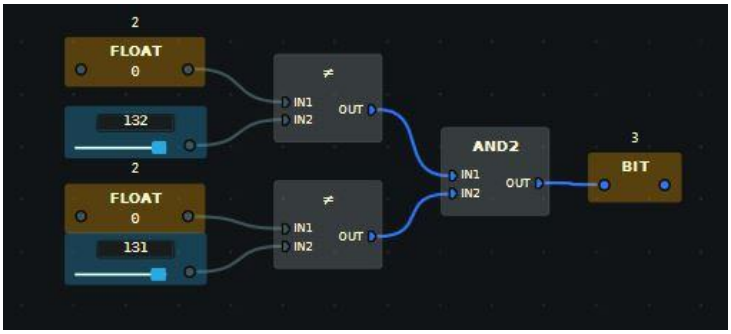


Figura 4-8 Diagrama para evitar accionamiento en la posición de equilibrio del joystick

Como el joystick por defecto en su posición central nos da una señal con valor 132 e incluso en determinados momentos de 131, para que no nos afecte a la programación realizando movimientos indeseados decidimos excluir esos valores tal y como se aprecia en la imagen.

Más adelante, una vez hayamos construido el diagrama para bajar las persianas, nos centraremos en la explicación sobre cuál es la utilidad del temporizador.

Movimiento de bajada

Las similitudes en el funcionamiento de este diagrama con respecto al anterior son muchas, habría que destacar algunos cambios significativos como pueden ser:

- Para que se accione el bloque de bajar la persiana la salida del bloque Arduino debe ser menor que el valor arbitrario de la constante.
- El bloque RS mantendrá en HIGH el Bit de la acción de bajar las persianas y un nuevo Bit de memoria auxiliar (4) que se utilizará en la realimentación del sistema.
- En este caso el Reset será activado cuando el sensor de posición de la persiana alcance el valor 0 ó se cumplan las condiciones del otro ramaje proveniente conjunción AND: Es decir, que se active el Bit 3 auxiliar descrito anteriormente, y que el nuevo movimiento del joystick sea contrario al efectuado previamente, en este caso hacia arriba.

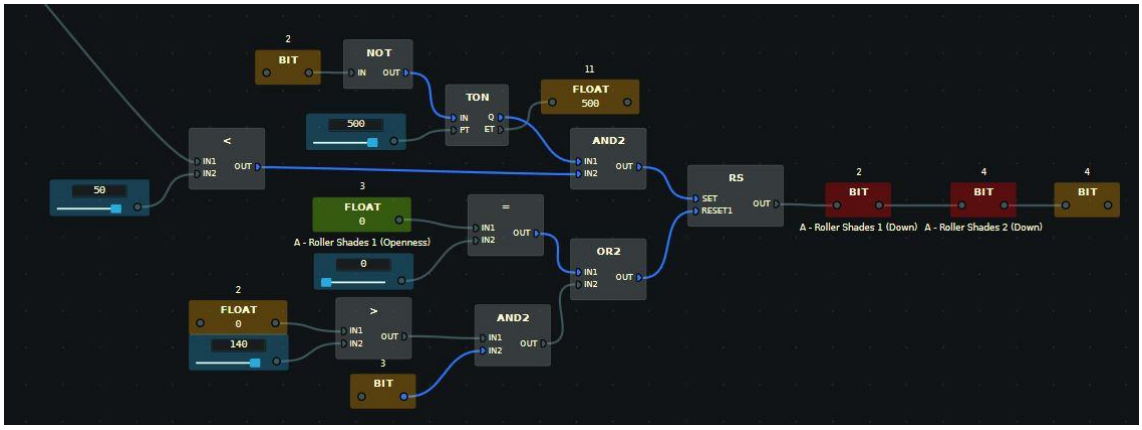


Figura 4-9 Diagrama del movimiento de bajada

Sincronización entre la subida, bajada y parada de la persiana

Llegados a este punto tenemos que sincronizar las órdenes para que todo funcione correctamente según vayamos moviendo el joystick se ejecuten los movimientos y las paradas de forma deseada.

Tras múltiples pruebas aplicando la metodología de prueba y error, llegamos a la conclusión que la mejor opción es añadir un temporizador o retardo de 500 milisegundos en ambos diagramas para evitar conflictos entre la señal emitida en primera instancia, ya sea de subida o baja, y la secundaria cuya misión es activar el Reset y cortar la alimentación de los bloques actuadores.

Para ello como se puede apreciar en las imágenes anteriores, usamos los dos bloques de memoria auxiliares (cuyos Bits eran 2 activado en la subida y 4 en la bajada) situados de forma cruzada, es decir, el Bit 4 activa el temporizador de la subida y el Bit 2 activando el temporizador de la bajada.

Este pequeño desfase de 500 milisegundos es esencial para evitar que una vez que movamos el joystick con la intención de detener la persiana a mitad de su recorrido, no comience a funcionar directamente en el sentido inverso, ya que por el temporizador no se cumpliría la condición del primer AND y lograríamos el cometido de la detención permanente.

De no aplicar esta distribución la persiana cambiaría de sentido en su movimiento pero no existiría una pausa permanente.

Este sería el esquema completo usado para el conjunto de las persianas de una de las orientaciones.

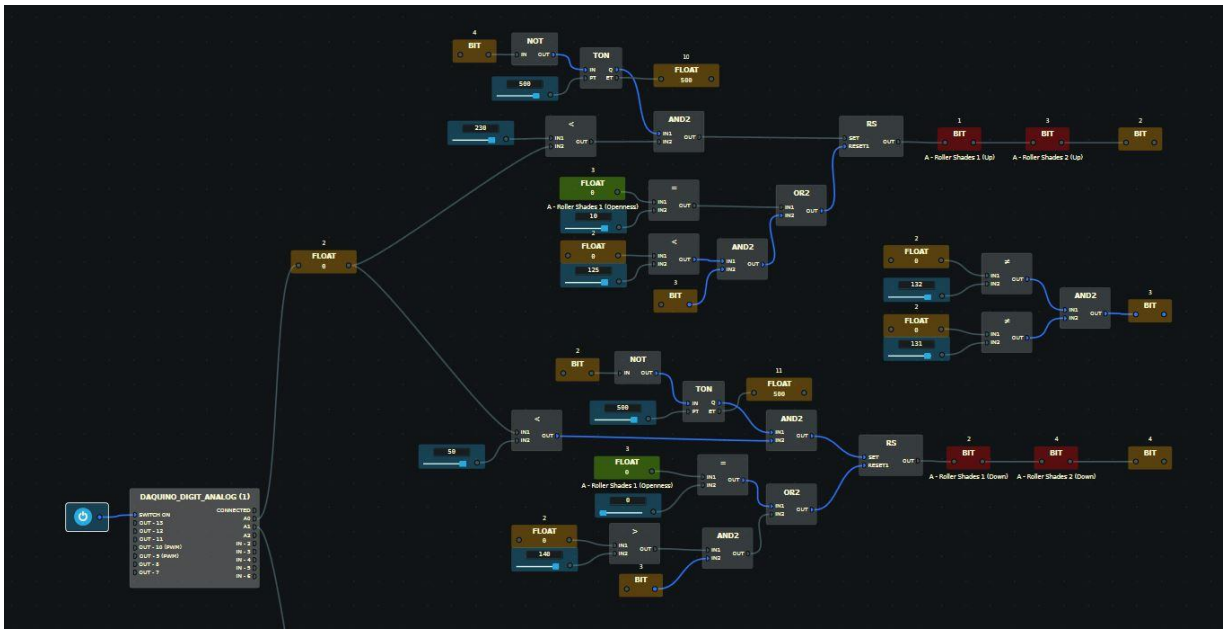


Figura 4-10 Programación del movimiento de subida y bajada de un conjunto de persianas

Siguiendo las mismas pautas explicadas con detenimiento para un conjunto de persianas, nos disponemos a diseñar el resto.

Ahora tendríamos que duplicar el diagrama y conectarlo a otro de los pines analógicos del Arduino donde vamos a conectar el otro eje del joystick. Posteriormente tendremos que realizar los pertinentes cambios en las variables que vayamos a usar ya que todas son distintas y deben tener otra dirección asociada a cada una de ellas.

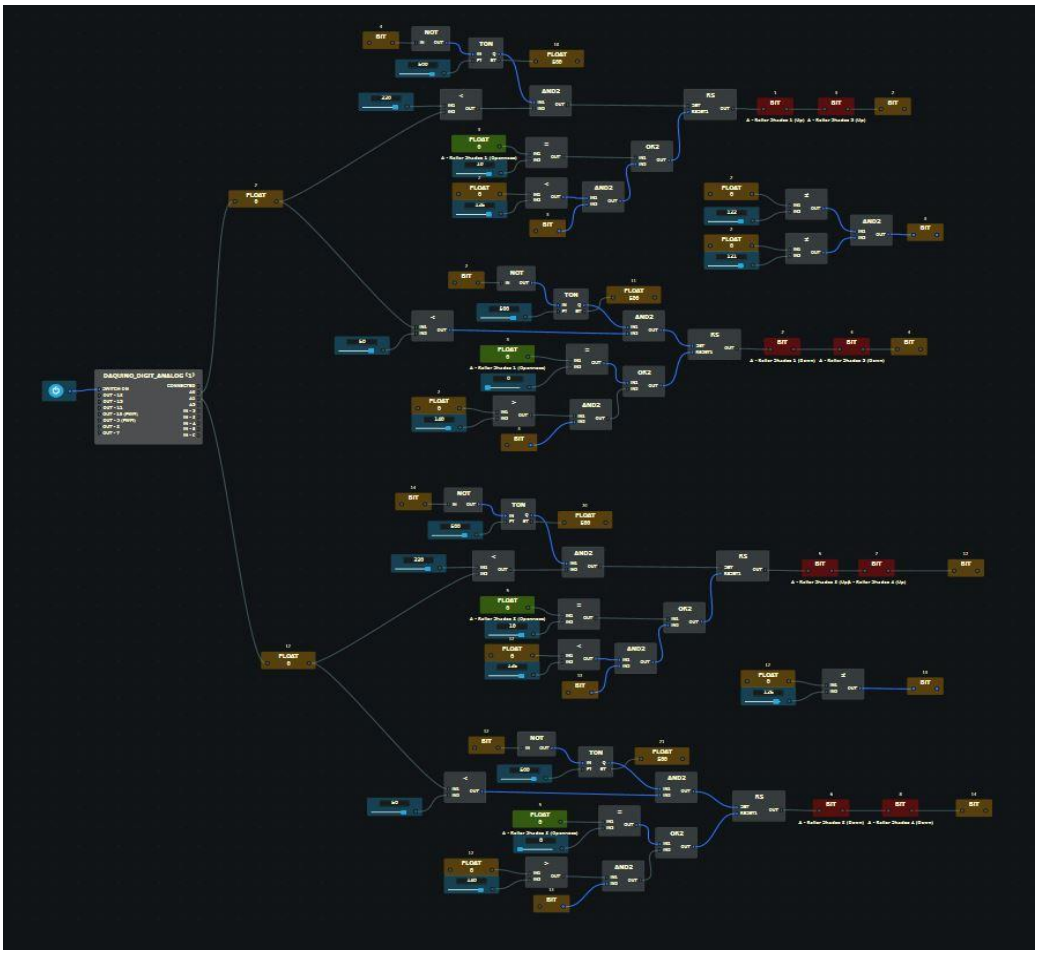


Figura 4-11 Programación de los dos conjuntos de persianas

Este resultaría el engorroso montaje final. Es importante destacar que tantas conexiones y multitud de condicionantes en el diagrama, lo hacen complejo y en muchas ocasiones da fallos o directamente no nos reconoce los movimientos del joystick. La solución más efectiva para todos ellos es desconectar y volver a activar la fuente de Bit que alimenta al bloque DAQUINO_DIGIT_ANALOG.

5 APLICACIÓN MEDIANTE DEMOSTRACIÓN DESAROLLADA: ELEMENTOS DIGITALES

5.1 Luz automática de la entrada

Como toma de contacto con los elementos digitales vamos a empezar con la configuración más simple de todas, la luz automática de la entrada de la casa.

Cuando el sensor de la entrada detecte la presencia de alguna persona y sea de noche, esto último lo reconocerá el sensor de brillo instalado en el jardín, automáticamente se encenderá la luz durante un tiempo establecido por nosotros mediante un temporizador.

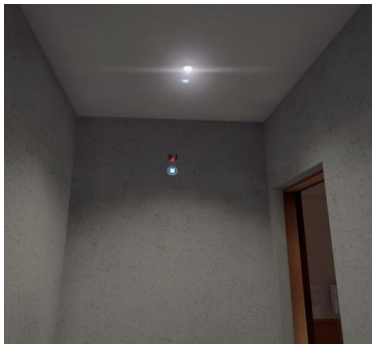


Figura 5-1 Elementos a activar en la entrada de la casa inteligente

Sensor de brillo situado a la derecha, justo a la izquierda vemos el altavoz para la sirena de la alarma. Ambos situados en el exterior de la vivienda junto a la puerta del garaje.

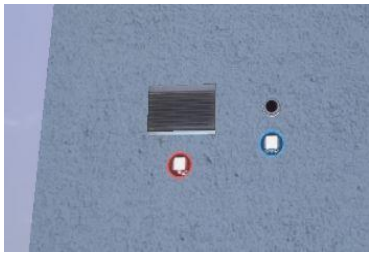


Figura 5-2 Sensor de brillo externo a activar

I/O Señales:

- Entradas:
  - Sensor de Movimiento de la entrada (Motion Detector). Bit 258.
  - Sensor de brillo del jardín (Brightness Sensor). Bit 259. Se encargará de medir los niveles de iluminación. Toma valor 1 si es de noche y 0 de día.

- Salidas:
  - Bloque activa luz entrada (Lights Entrance). Bit 191.

- Otras funciones:
  - AND, SR, TON y fuente numérica.
  - Bloque memoria auxiliar (R Luz entrada). Bit 1. Más adelante por conflictos con otros bloques tendremos que modificar esa dirección. Se activará una vez que el temporizador encienda su salida.

Explicación de la solución

Si se cumple que los nodos del sensor de movimiento situado en la entrada de la casa y del sensor de brillo se ponen a 1, se activará el bloque de encendido de las luces y se mantendrá la señal en HIGH durante un cierto tiempo debido al bloque Set Reset (SR) y al temporizador que está situado justo a continuación. Cuando se cumpla el tiempo establecido por la fuente numérica, en este caso 10 segundos, se pondrá a 1 el bloque de memoria Bit 1 auxiliar y se encargará de realimentar el sistema y activar el Reset, apagando así la luz.

Nótese que la aplicación de la condición del sensor de brillo es una medida de cara al ahorro energético, ya que de no existir la luz se encendería indiferentemente fuera de día o de noche.

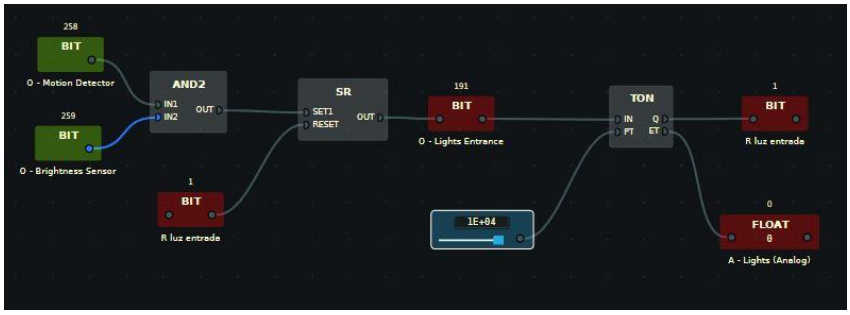


Figura 5-3 Diagrama de bloques de la luz automática de la entrada

5.2 Mando a distancia programmable

En los diagramas de bloques siguientes vamos a utilizar un mando a distancia virtual que nos proporciona el software HOME I/O. Éste nos permitirá manipular distintos elementos de la casa inteligente una vez programados por nosotros mismos.

Para ello es necesario cambiar su estado a programable en CONNECT I/O mediante el botón situado en la esquina superior derecha del mismo, tal y como se aprecia en la siguiente imagen:

Figura 5-4 Mando virtual programable

5.3 Luces del jardín y porche

I/O Señales:

- Entradas:
  - Botón 4 (Remote button 4). Bit 277.
  - Sensor de brillo (Brightness Sensor). Bit 259. Toma valor 1 si es de noche y 0 de día.
  -
- Salidas:
  - Bloque que enciende la luz de la piscina (Light pool). Bit 189.
  - Bloques que activan las luces del porche (Light Porch 1 y 2). Bits 187 y 188.

- Otras funciones:
  - AND, OR, RS y FTRIG.
  - Bloque memoria (Aux porche). Bit 444. Será la variable que utilizaremos para cerrar el circuito y conseguir la realimentación del mismo para el correcto funcionamiento del encendido y apagado de la luz.

Figura 5-5 Luces del porche y piscina vistas desde el exterior

Explicación de la solución

El diagrama es similar al montaje anterior. En este caso las condiciones iniciales que se tienen que cumplir son: que sea de noche también, es decir, que el sensor de brillo esté activado y además que hayamos pulsado el botón 4.

El uso del nodo trigger de caída (FTRIG) nos permitirá accionar la salida del esquema con sólo pulsar y soltar.

De esta forma activaremos los bloques actuadores de las luces de la piscina, y los dos conjuntos de luces del porche.

Para apagar las luces tenemos que enviar la señal Reset al bloque RS. Para ello, tiene que haberse activado el nodo auxiliar y además haber vuelto a pulsar el botón 4 del mando. La otra opción que automáticamente nos apagaría la luz sería que se desactivara el sensor de brillo, es decir, que detectara un nivel de luminosidad diurno.

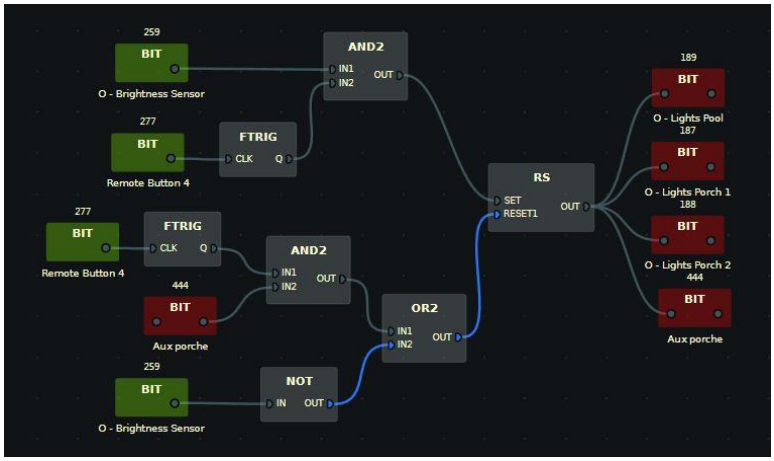


Figura 5-6 Diagrama resultante del accionamiento de las luces del porche y piscina

Luces del jardín

I/O Señales:

- Entradas:
  - Botón 3 (Remote button 3). Bit 276.
  - Sensor de brillo del jardín (Brightness Sensor). Bit 259. Toma valor 1 si es de noche y 0 de día.

- Salidas:
  - Bloque que activa las luces del jardín (Lights Garden). Bit 150.

- Otras funciones:
  - AND, OR, RS, FTRIG y Temporizador TON.
  - Bloque de memoria (Aux luces jardín). Bit 3. Nodo usado para realimentar el bloque RS.

La programación es análoga a la de las luces del porche y de la piscina. Se pretende controlar las luces del jardín principal para poder encenderlas y apagarlas mediante el botón 3 del mando a distancia. También tendremos en cuenta si es de día o de noche gracias al sensor de brillo.

Al igual que con todos los elementos tendremos que activar previamente el estado de programación. En la siguiente captura se observa la posición de dicho interruptor de estados y los focos en sí que se desean manipular:

Figura 5-7 Focos externos del jardín

Explicación de la solución

La salida se accionará cuando se pulse y suelte (gracias al trigger de caída, FTRIG) el botón 3 del mando inalámbrico y que el sensor de brillo alimente a la puerta lógica AND, es decir que sea de noche.

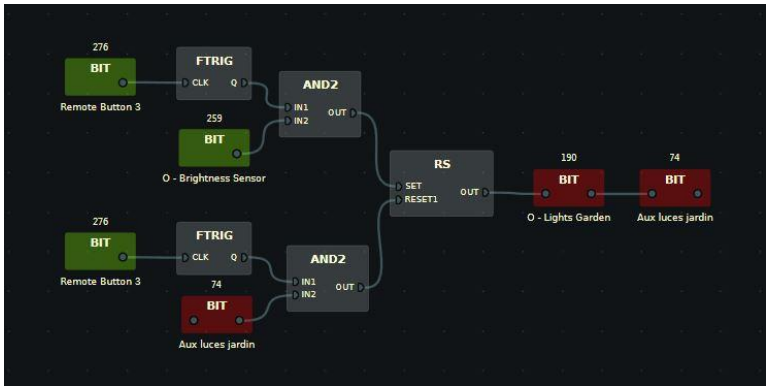


Figura 5-8 Diagrama de bloques del accionamiento de las luces del jardín

La variable Aux Luces del jardín será el nodo encargado de realimentar el bloque RS. Si se cumple que se vuelve a pulsar el botón 3 y la memoria ya estaba activa, se mandará un Reset y apagará las luces.

5.4 Alarma de seguridad

Nos disponemos a llevar a cabo la creación del sistema de seguridad en caso de intruso en la vivienda. Para ello nos centraremos únicamente en los accesos que tienen lugar por la puerta principal, garaje y salón accediendo desde el jardín.

I/O Señales:

- Entradas:
  - Sensor de movimiento (Motion detector). Serán bloques tipo Bit con las direcciones 15,80 y 82. Se activan cuando detectan movimiento en sus cercanías.
  - Sensores de apertura en las puertas (Door detector). Bits 13, 78 y 79. Tienen valor 1 si las puertas se encuentran cerradas.
  - Alarma armada (Alarm Keypad- Armed). Bit 82. Nos informa que la alarma ya ha sido previamente armada o activada.
  - Acción de desarmar (Alarm Keypad Disarm). Bit 60. Desactiva la seguridad.
  - Botón 8 (Remote Button 8). Bit 281. Botón del mando virtual que controlará la activación/ desactivación de la seguridad.
- Salidas:

- Sirena (Siren). Bit 58. Cuando se activa comienza a sonar la sirena de la casa.
- Acción de armar alarma (Alarm Keypad Arm). Bit 59. Activa la seguridad de la vivienda.

- Funciones:

  - Bit tipo memoria auxiliar. 96. Variable que será empleada como bandera.
  - Bloques de lógicos: RS, Ftrig, OR2, AND, NOT, Fuente Bit, etc.

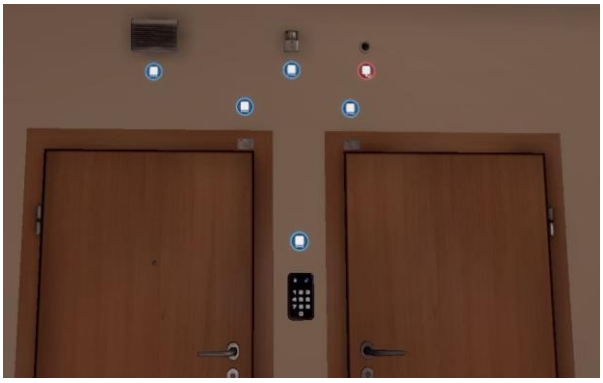


Figura 5-9 Sensores y actuadores de la alarma

Explicación de la Solución

La idea del montaje consiste en que una vez que pulsemos el botón 8 del mando virtual se lleve a cabo la acción de armar la seguridad de la vivienda y si volvemos a pulsar el mismo botón nos la desactive e incluso nos apague la sirena en el supuesto de estar sonando por intrusión. Una vez activada la alarma dicha sirena además puede comenzar a sonar también por apertura de las puertas protegidas así como por presencia detectada en el interior de la casa.

A continuación veremos el desarrollo del diagrama que nos permite activar la alarma a partir del mando inalámbrico virtual. Como viene siendo habitual, para la programación de los pulsadores del mando recurriremos a un Trigger, en este caso de bajada.

Nótese el uso del bit auxiliar con dirección 96 cuya misión es primordial ya que será el que nos marque si ya hemos pulsado o no el botón anteriormente para activar la alarma, y así efectuar de manera correcta el apagado de la misma. Es decir, si la alarma ha sido armada, ese bit se pondrá a 1, de forma que en cuanto volvamos a pulsar el botón 8 mandaremos una señal al RESET apagando el sistema gracias al nodo de la puerta lógica AND.

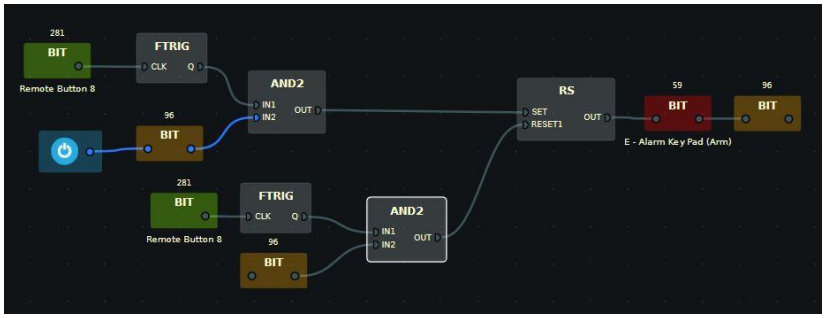


Figura 5-10 Diagrama de accionamiento de la alarma

En este panel podremos visualizar gráficamente si la alarma se encuentra activada o desactivada mediante los dos iconos. En el momento que comencemos a programar en CONNECT I/O el teclado quedaría inservible, con lo cual no nos ofrecerá ninguna utilidad.

Figura 5-11 Panel de alarma

El objetivo de la segunda parte del montaje es construir la activación de la sirena. Esta acción tendrá lugar si se da algunas de las siguientes situaciones una vez activada la alarma:

- Los sensores de movimiento detectan alguna anomalía en sus proximidades.
- Si las puertas que se pretenden proteger, se abren de forma imprevista. Como ya se explicó antes, estando abiertas sus sensores tomarán valor 0 mientras que el valor de estos sensores con las puertas cerradas es de 1. Por ese motivo hemos empleado un bloque NOT para usar la lógica inversa.

Si se da cualquiera de estas circunstancias y el bit de alarma activada esta a 1 comenzara a sonar la sirena de forma ininterrumpida por el SET hasta que volvamos a pulsar el botón 8 y se active el RESET. En ese momento se efectuará el desarme y el apagado de la sirena de manera simultánea, la alarma será desactivada.

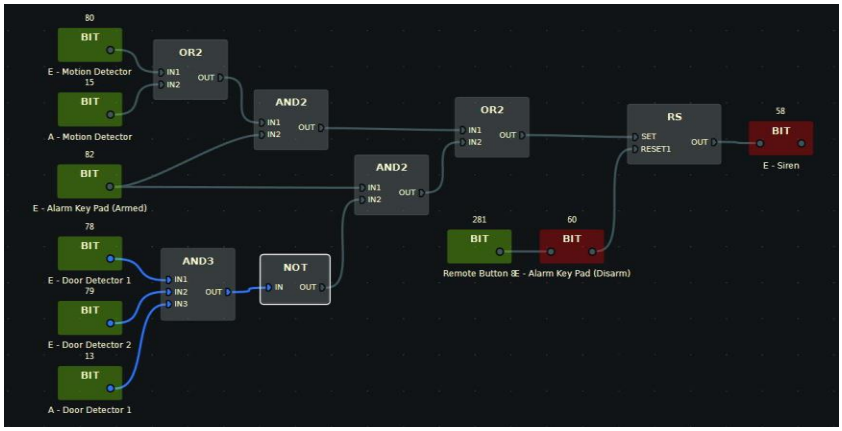


Figura 5-12 Diagrama de activación de la sirena en caso de intrusión

5.5 Automatización de la puerta del garaje y sincronización con las luces

En este apartado del proyecto realizaremos la configuración automática de la puerta plegable del garaje con su respectiva sincronización de las luces del interior del mismo cuando sea de noche.

La programación consiste en que una vez que pulsemos el botón del mando inalámbrico, el motor que nos acciona la apertura de la puerta se encienda independientemente de si estamos dentro o fuera del garaje, haga una pausa durante unos segundos manteniendo la puerta completamente abierta, para luego volverse a encender y que nos cierre definitivamente la puerta. Todo el procedimiento irá acompañado de la información que nos ofrecerán varios sensores situados en distintos puntos del recorrido de la puerta.

A su vez todo este proceso irá sincronizado con las luces del interior del garaje siempre y cuando sea necesario. Es decir, el sensor de brillo del interior determinará si se requiere encender las luces en función de los niveles de iluminación analizados una vez que la puerta esté abierta.

Además añadiremos algunas condiciones más que iremos desmenuzando con mayor detenimiento a medida que expliquemos la solución. Por ejemplo el sensor por infrarrojos situado en la puerta para detectar si hay algún obstáculo impidiendo el cierre del portón.

En las imágenes siguientes podemos apreciar con claridad los elementos los cuales habría que modificar su estado de programación tal y cómo venimos haciendo en apartados anteriores.

Elementos relacionados con las luces interiores y sensores: En la pared frontal a la puerta del garaje encontramos el sensor de movimiento y el de brillo; mientras que en el lateral junto a la puerta de acceso al hoole de la vivienda se sitúan los interruptores de la luz del distribuidor y del garaje.

Figura 5-13 Motor de la puerta y sensores del garaje

Vista desde el interior. Observamos el motor en el techo y el interruptor de accionamiento de la puerta del garaje junto a ella en el lateral derecho.

Figura 5-14 Vista interna de la puerta del garaje

Vista desde el exterior. Podemos identificar en los laterales de la puerta los sensores infrarrojos que se mencionaron anteriormente, aunque no requieren de activación puesto que el nodo ya viene incluido en el conjunto del estado de programación del motor.

Figura 5-15 Vista externa del garaje

Vamos a dividir el análisis de la sección en dos partes: Diseño del diagrama de bloques del accionamiento de la puerta del garaje y posteriormente las luces con su sincronización.

1) Accionamiento de la puerta del garaje

I/O Señales:

- Entradas:
  - Botón 1 del mando inalámbrico (Remote Button 1). Bit 274. Encargado de accionar el motor una vez pulsado.
  - Interruptor de subida y bajada (Switch up/down). Bit 94. Interruptor situado en el lateral derecho como se aprecia en la última imagen que nos permite abrir manualmente el portón.
  - Sensor de posición de la puerta del garaje abierta (Garage Door Opened). Bit 100. Se activa cuando la puerta del garaje se abre completamente.
  - Sensor de posición de la puerta cerrada (Garage Door Closed). Bit 101. Este sensor se activa cuando el portón se encuentra cerrado.
  - Sensor infrarrojo de la puerta (Garage Door Infrarred). Bit 102. Sensor que se activa única y exclusivamente el tiempo en el que un obstáculo se sitúa en medio del acceso impidiendo que la señal emitida por el transistor llegue al receptor infrarrojo virtual.

- Salidas:
  - Acción de abrir garaje (Garage Door Open). Bit 22. La activación de este nodo genera el accionamiento del motor del garaje para abrir el portón.
  - Acción de cerrar garaje (Garage Door Close). Bit 23. La activación de este otro nodo genera el accionamiento del motor del garaje para cerrar el portón.

- Otras funciones:
  - AND de 2 y 3 entradas, OR, NOT, RS, TON, Fuentes numéricas,
  - Bloque auxiliar llamado “Variable aux bajada”. Bit 281. Se activará mientras la puerta se esté cerrando. Nos permitirá realizar bucles e interrumpir dicha bajada.
  - Bloque auxiliar para abreviar el diagrama llamado “Pulsar botón 1 ó Switch”. Bit 279. Debido a que los nodos de pulsar el botón 1 o el interruptor de encendido se encuentran unidos entre sí mediante un bloque OR y serán utilizados en varios puntos, para compactar el diagrama se ha optado por utilizar un nodo a modo de enlace. De esta forma, además de hacer más atractivo visualmente el diagrama, lo haremos más legible.

- Nodos medidores usados en temporizadores: “T.puerta garaje abierta. Float 4. Permitirán al usuario poder controlar visualmente el tiempo que lleva funcionando cada temporizador.

Explicación de la solución:

La puerta se abrirá si y sólo si pulsamos el botón 1 del mando inalámbrico o si se da la circunstancia de que el nodo del sensor infrarrojo de la puerta se activa, es decir, es tapado por algún obstáculo y la puerta en ese preciso instante se está cerrando. Cuando se dé una de estas dos situaciones la salida del bloque RS se pondrá de forma permanente en HIGH hasta que se active la señal Reset. Esta última se activará como resultado del sensor de posición de la puerta al llegar arriba.

Una vez que el portón se situe en su posición superior, la puesta a 1 del sensor de apertura activará un temporizador tipo TON durante un tiempo impuesto por nosotros, en este caso 10 segundos. La salida Q activará una señal Set y esta alimentará al bloque de memoria auxiliar “variable aux bajada”, el cual se utiliza para realimentar el circuito en la acción de subir la puerta como condición adicional en la activación del sensor infrarrojo. Si la variable auxiliar está a uno, puerta bajando, y se activa el sensor de infrarrojos, la puerta debe abrirse por existir un obstáculo en el acceso.

El Reset dependerá de si volvemos a pulsar el botón del mando o si la puerta se ha cerrado completamente.

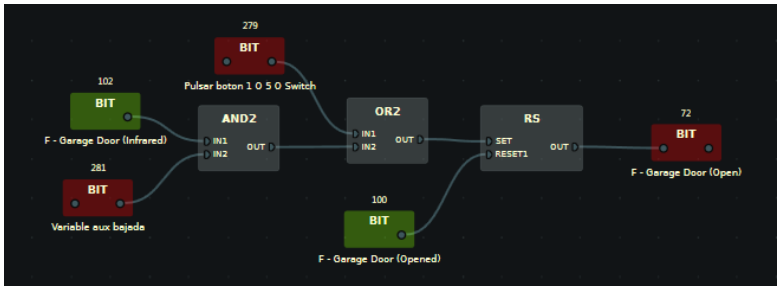


Figura 5-16 Diagrama de apertura de la puerta del garaje

La otra alternativa está pensada para el cierre de la puerta automáticamente tras haber nosotros entrado en el interior del garaje. Esto es posible gracias a los sensores de movimiento, de posición de apertura de la puerta y el de infrarrojos. Para mantener a uno la señal una vez pasado por delante del sensor de infrarrojos tendrá que usarse de nuevo un bloque RS, ya que la señal de este bloque únicamente esta activada en el momento del obstáculo. A diferencia de este, es mportante conocer también que el sensor de movimiento permanece activo varios segundos tras detectar movimiento.

Si se cumple que todos estos sensores se activan de forma consecutiva en un pequeño intervalo de tiempo, quiere decir que hemos entrado en el interior del garaje. A partir de este momento se activará el nodo de cierre de la puerta.

La condición del Reset de este segundo bloque RS dependerá de que se haya vuelto a pulsar el botón del mando virtual o interruptor, que la puerta se haya cerrado completamente o se haya vuelto a activar el sensor de infrarrojos.

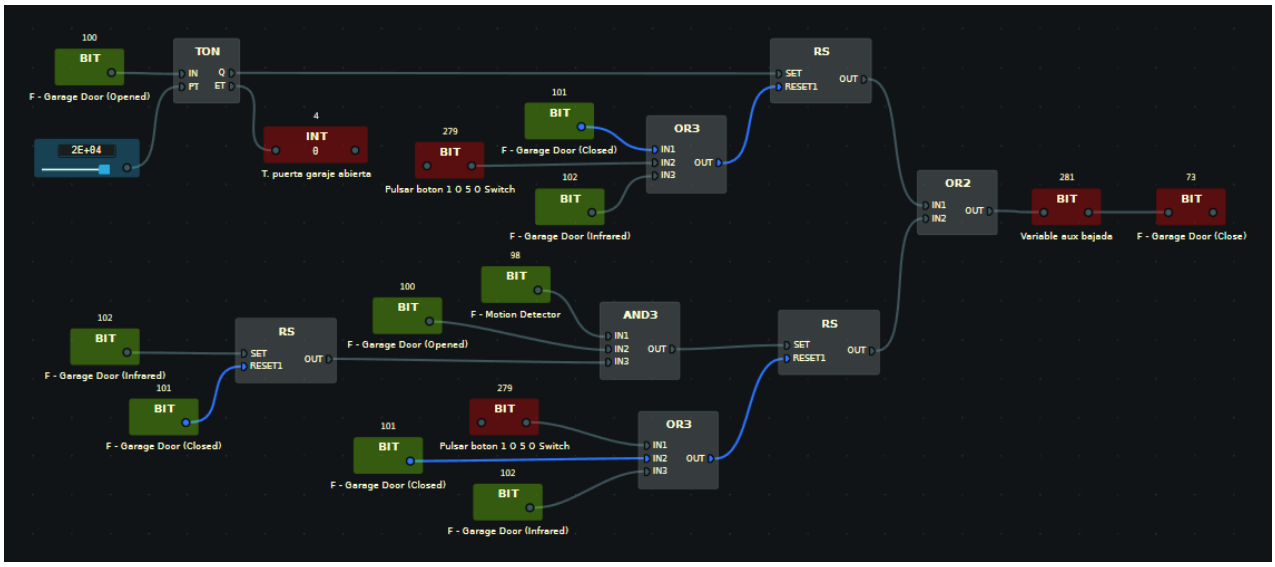


Figura 5-17 Diagrama de cierre de la puerta del garaje

A continuación mostramos cuales son los nodos que componen el bloque de memoria “Pulsar botón 1 o switch” (Bit 279) que se utilizado como enlace para abreviar el esquema:

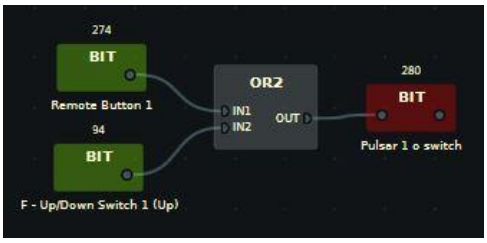


Figura 5-18 Nodo enlace

2) Luces y sincronización

I/O Señales:

- Entradas:
  - Sensor de brillo del jardín (Brightness Sensor). Bit 99.
  - Sensor de movimiento del interior del garaje (Motion detector). Bit 98.
  - Acción de abrir la puerta del garaje (Garage Door Open). Bit 72.
  - Interruptor de la luz del recibidor (Light switch 1). Bit 92. Interruptor de tipo actuante.
  - Interruptor de la luz del interior del garaje (Light switch 2). Bit 93. Interruptor de tipo pulsante o momentáneo.

- Salidas:
  - Acción de encender la luz del recibidor (Light 1). Bit 69.
  - Acción de encender la luz del garaje (Light 2). Bit 68.

- Otras funciones:
  - AND, OR, RS, TON, FTRIG y Fuentes numéricas.

- Nodo auxiliar llamado “Aux switch luces garaje”. Bit 17. Será la variable que permita apagar la luz del interior garaje de forma manual a partir del interruptor pulsante cuando la luz haya sido anteriormente encendida.

Explicación de la solución:

Las salidas del diagrama de bloques serán el nodo que enciende la luz del recibidor, nodo Lights 2, que se encenderá mediante un interruptor tipo actuante situado en el recibidor, y el nodo Lights 1, es decir, la luz interior del garaje accionada por un interruptor de tipo pulsador.

Esta diferencia en la configuración podemos apreciarla en la complejidad del diagrama de bloques de la segunda salida.

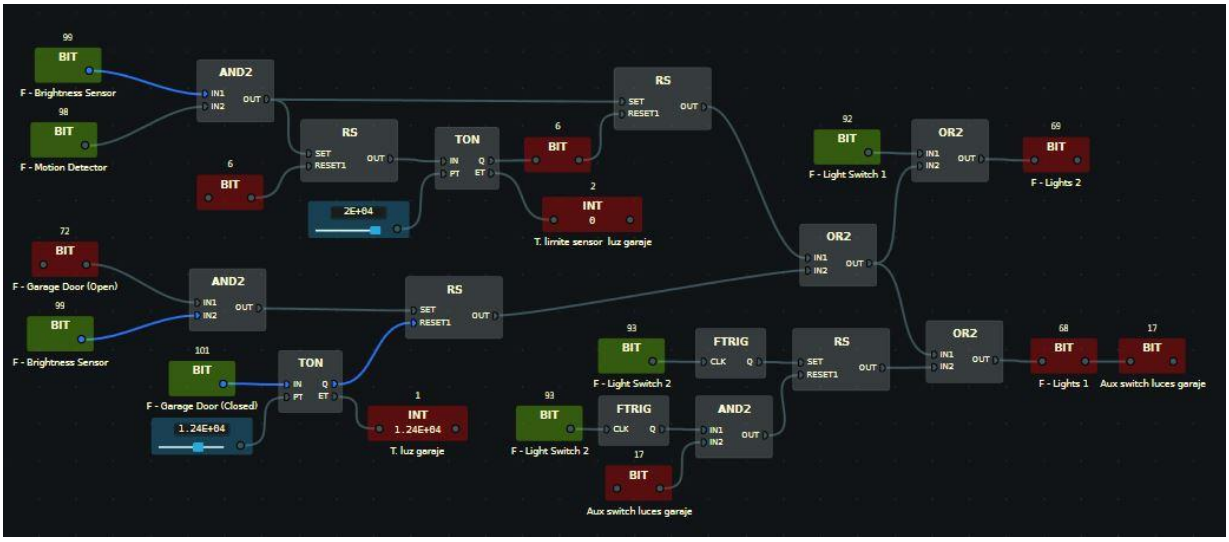


Figura 5-19 Diagrama de bloques sincronizado con las luces del garaje

Estos nodos de salida pueden también ser accionados de manera simultánea y automática haciendo uso de los distintos sensores que tenemos en el garaje: sensor de movimiento, de brillo y de la puerta del garaje.

Dentro de esta posibilidad, podemos encontrar otras dos variantes:

- En el momento que se accione el motor de la puerta del garaje, comience a abrirse y se cumpla que el sensor de brillo esté activado, las luces del interior, es decir, las del recibidor y del garaje en sí, deben mantenerse encendidas por el efecto de la señal SET de un bloque RS.
  Un temporizador comenzará a contar en el momento en que la puerta se vuelva a desplegar completamente y se active el sensor de cierre de la puerta. Cumplido el tiempo del temporizador se enviará un RESET y las luces se apagarán.

- Si se enciende el sensor de movimiento y también está activado el sensor de brillo interior del garaje, mediante otro bloque RS se activará una señal de SET que encenderá el conjunto de luces.
  Una vez tenga lugar la acción, otro temporizador comenzará a contar. Cuando termine el tiempo se enviará la señal de Reset y cortará inmediatamente la alimentación al nodo de las luces.

5.6 Reja automática

Al igual que con la puerta del garaje, vamos a programar la reja externa para que pulsando un botón del mando inalámbrico se lleve a cabo su apertura, permanezca abierta completamente y pasado un tiempo se vuelva a cerrar.

Añadiremos algunas condiciones como apertura obligatoria de la reja en caso de obstáculo, sincronización de la reja con las luces del jardín en caso de poca iluminación y con la puerta del garaje para la entrada de vehículos, etc.

En la siguiente captura vemos donde se sitúa el motor que acciona la apertura/cierre de la reja así como donde se encuentra el botón de cambio de estado, ya preparado para programar por su color azul.

Figura 5-20 Vista desde el interior de la reja y su motor

Además también podemos apreciar la posición de los distintos sensores por infrarrojos de la puerta: Uno para determinar la presencia de un obstáculo justo en el paso por la reja, otro para informar acerca de la apertura cuando llegue a su posición final, y por último un tercero que no nos va a aportar ninguna utilidad y por ese motivo no vamos a emplear.

En esta imagen podemos ver lo descrito hasta el momento desde una vista externa a la reja de la vivienda:

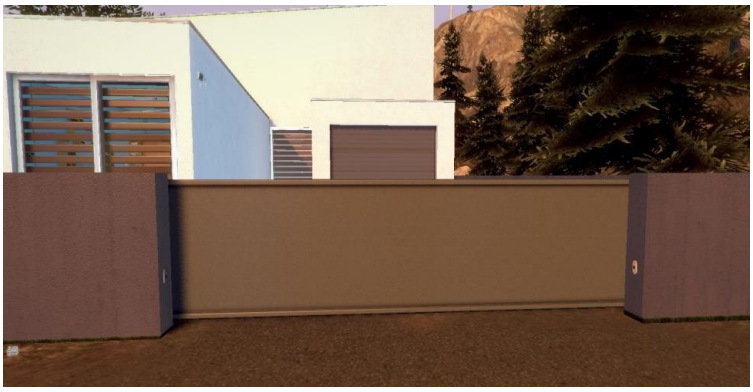


Figura 5-21 Vista de la reja desde el exterior de la casa inteligente

I/O Señales:

- Entradas:
    - Botón 5 del mando inalámbrico (Remote Button 5). Bit 278.

  - Sensor de posición de la reja que se activa cuando se encuentra abierta completamente (Entrance Gate Opened). Bit 260.
  - Sensores infrarrojos de la reja (Entrance Gate Infrarred 1 y 2). Bits 262 y 263.
  - Sensor de posición de la reja que se activa cuando se encuentra cerrada (Entrance Gate Closed). Bit 261.

- Salidas:
  - Acción de abrir reja (Entrance Gate Open). Bit 193.
  - Acción de cerrar reja (Entrance Gate Close). Bit 194.

- Otras funciones:
  - AND de 2 y 3 entradas, OR, NOT, RS, TON, Fuentes numéricas,
  - Bloque auxiliar llamado "Aux cierre". Bit 260.
  - Bloques medidores usados en temporizadores. Float 71 y 72.

Explicación de la solución

La reja se abrirá gracias a la acción permanente de la salida del bloque RS, que se activará si se cumple una de las siguientes condiciones:

1. Pulsamos el botón 5 del mando inalámbrico.

2. Estemos saliendo o entrando del reciento limitado por la reja de la vivienda y se haya activado uno de los sensores infrarrojos de la reja, ya sea el externo o el interno; además que la variable auxiliar "Aux cierre" esté activa, lo cual implica que la reja se está cerrando; y que el sensor de apertura esté apagado, es decir que la puerta no esté abierta.
   Como el sensor de la reja abierta queremos que tome valor uno en todo momento excepto cuando la reja esté abierta, tenemos que poner a su salida un nodo de negación de tipo Not. Así evitaremos el accionamiento del motor de la reja si se activan los sensores por infrarrojos y la variable auxiliar de cierre.

Todo el mecanismo descrito hasta ahora se puede comprender con más facilidad visualizándolo en el diagrama, exactamente en la rama superior izquierda:

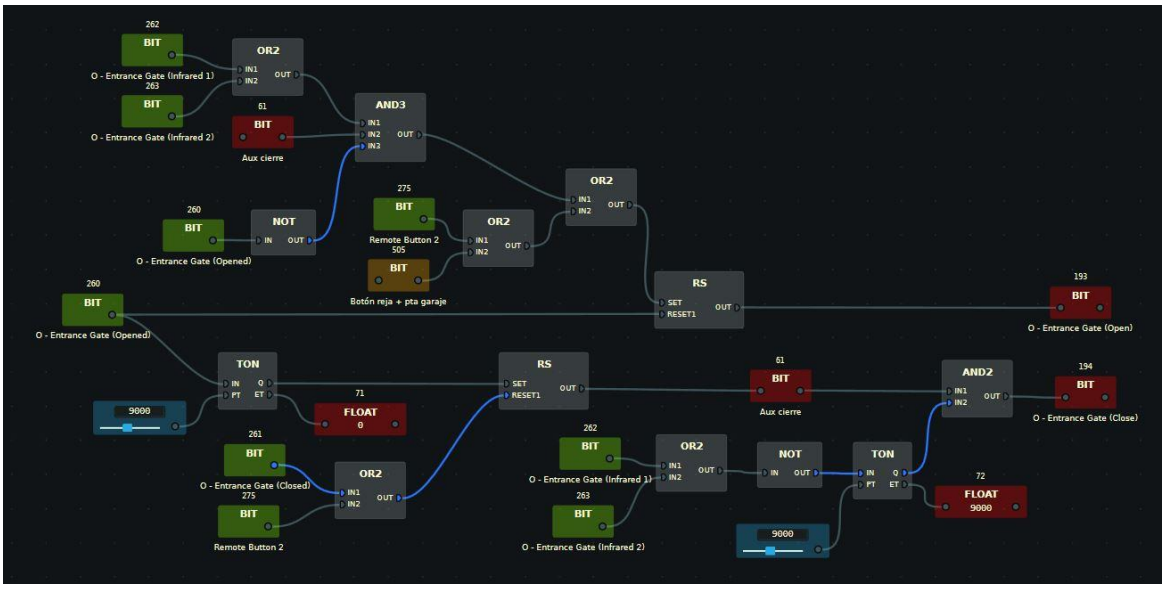


Figura 5-22 Diagrama de bloques del funcionamiento de la reja automática

En cuanto la puerta se abra completamente y sea reconocido por el sensor de apertura, se activará un temporizador de 5 segundos el cual a su salida activará a otro nodo del tipo RS. Dicho nodo alimentará al bloque auxiliar denominado como "Aux cierre" que conecta a su vez con el bloque que cierra la reja. La condición que activará el Reset que anula el cierre de la reja será pulsar el botón 5 del mando o que la puerta se cierre y lo detecte el sensor de recorrido.

Por último y completando el bucle descrito anteriormente en la ejecución de la apertura de la puerta del garaje, si se activa alguno de los sensores por infrarrojos situados en los laterales del acceso, debido a la acción de una función AND se producirá una interferencia. Dicha interferencia cortará la alimentación al bloque de cerrar la reja y automáticamente se procederá a realizar de nuevo un ciclo de apertura de la reja.

Cabe destacar la presencia de otro nodo de negación para lograr que la salida que está conectada a la función AND siempre esté activa y en el momento que exista una interferencia o fallo, este bloque cambie su estado y genere un corte de la conexión.

5.7 Sincronizacion de la reja automática, puerta del garaje y luces del jardín

Tras la programación de forma individual de cada uno de los elementos tecnológicos que deseamos controlar, nos disponemos a construir la comunicación entre todos ellos. La misión de este apartado es dotar a nuestra casa inteligente de mayores comodidades y cualidades domóticas si cabe, incrementando las calidades de vida así como dar más realismo a la simulación.

Para ello tendremos que decidir en primer lugar cuál es la meta buscada, qué es más adecuado y cómo lo haremos buscando siempre la opción más práctica posible pensando en el usuario y en la eficiencia o consumo energético de la casa.

En el apartado donde se programó la puerta del garaje, se realizó directamente la comunicación con sus luces interiores. En este caso se va a proceder a la construcción del análogo entre las luces del jardín con la reja externa.

Además los sensores de brillo controlarán los niveles de iluminación durante las distintas horas del día y determinarán si será necesario encender los focos del jardín en caso de apertura de la reja. De esta forma, siempre nos moveremos dentro de un marco de bajo consumo.

Por último, se programará la apertura simultánea de la reja externa con la puerta del garaje, ambas temporizadas, pulsando un botón del mando inalámbrico. Así, en caso de entrada de un vehículo podríamos

acceder a la casa inteligente cruzando en primera instancia la reja, para luego posteriormente y totalmente coordinada en tiempos, la puerta del garaje sin tener que accionar nada más.

Todo este procedimiento iría acompañado del encendido de las luces internas y externas si los niveles de iluminación ofrecidos por los sensores de brillo en ese momento así lo requieren.

En la siguiente imagen observamos la maniobra programada en pleno proceso de ejecución:

Figura 5-23 Vista desde el exterior en el momento de la sincronización completa

I/O Señales:

Para este apartado no se añadirán nuevos nodos que sean necesarios definir.

Explicación de la solución

Comenzaremos con la comunicación entre las luces del jardín y la reja de la casa.

La salida de este esquema, es decir la activación de las luces del jardín, se llevará a cabo siempre y cuando se cumplan las condiciones que preceden a los dos bloques RS:

- Se pulse y suelte (gracias al trigger de caída) el botón 3 del mando inalámbrico y por supuesto que el sensor de brillo tenga valor 1, es decir que sea de noche.
- Se haya accionado el bloque que abre la reja externa de la casa inteligente y el sensor de brillo.

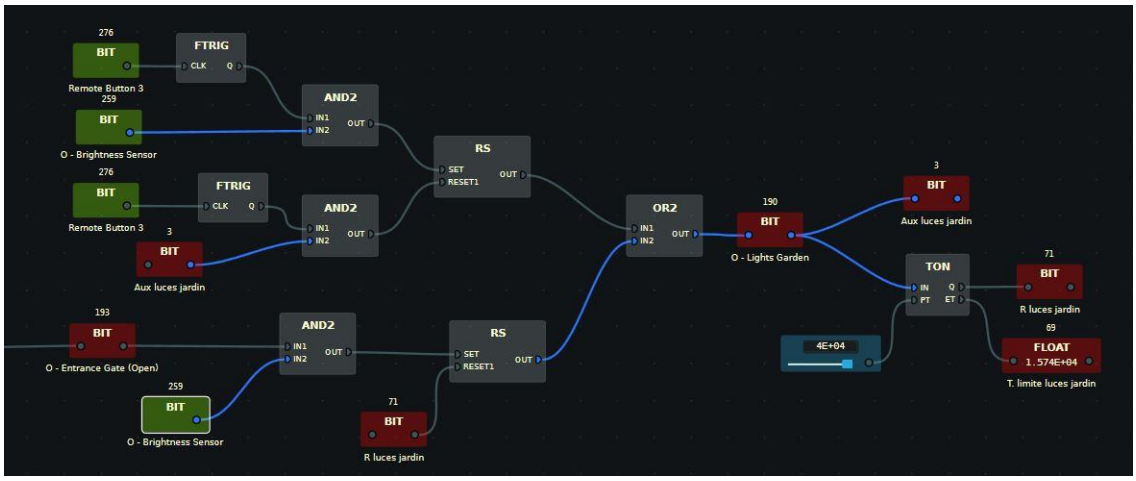


Figura 5-24 Modificación del diagrama de bloques del accionamiento de las luces del jardín

Como consecuencia de la activación de la salida, se pondrán en HIGH otras dos variables:

- Aux Luces del jardín. Será el nodo usado para realimentar el bloque RS descrito anteriormente. Una vez que se vuelva a pulsar el botón 3 y dicho bloque siga activo se mandará una señal Reset que apagará las luces.
- R luces del jardín. Tras activarse la salida Q del temporizador de tipo TON, se pondrá a 1 esta variable y apagará la iluminación. Este tiempo ha sido escogido minuciosamente para que en caso de entrar con algún tipo de vehículo al interior del reciento no nos quedemos a oscuras.

Siguiendo la línea de trabajo, ahora tendremos que hacer las modificaciones pertinentes para la configuración de la apertura simultánea de la puerta del garaje y la reja externa.

Simplemente tendremos que añadir la entrada Remote button 2 al diagrama auxiliar de la puerta del garaje. Acompañando así a la entrada Remote button 1, ambos enlazados mediante una puerta OR. De esta forma, cuando se pulse el botón 1 se abrirá solamente la puerta del garaje mientras que cuando se pulse el botón 2 tendrá lugar el proceso explicado.

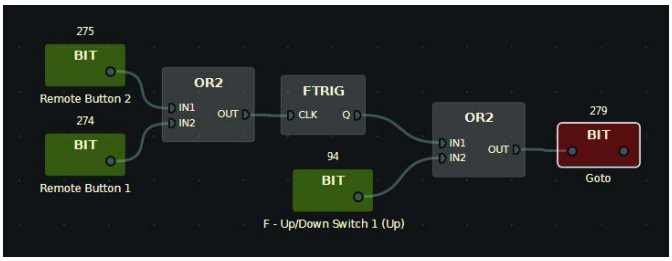


Figura 5-25 Diagrama simplificado de la sincronización

6 RESULTADOS Y CONFIGURACIÓN FINAL

La técnica divide y vencerás consiste en descomponer el problema en un conjunto de subproblemas más pequeños. Después se resuelven estos subproblemas y se combinan las soluciones para obtener la solución para el problema original.

Básicamente esta es la estrategia seguida hasta este punto del proyecto tras elaborar la programación de los distintos elementos tecnológicos y las acciones que deseamos que se lleven a cabo.

Nuestra meta ahora es relacionarlos y construir un soporte real que nos permita centralizar el control de la casa inteligente, haciéndo compatibles todos los mecanismos entre sí y distinguiéndolos en función de la naturaleza de su control: Analógico y digital.

6.1 Configuración analógica

Dado que se nos ofrece la posibilidad de elegir entre dos plugins para conectar el hardware Arduino al software CONNECT I/O (Véase capítulo 4 del proyecto, comunicación entre Arduino y CONNECT I/O) tenemos que escoger el nodo DAQUINO_DIGIT_ANALOG puesto que es el bloque que posee salidas analógicas.

La placa Arduino tiene hasta 6 pines analógicos: desde A0 hasta A5. En el proyecto tendremos que conformarnos con tres puertos única y exclusivamente para trabajar en el control de los distintos elementos. Son las siguientes tres salidas analógicas: A0, A1 y A2. Como ya sabemos, las salidas analógicas del plugin son entradas analógicas de la placa Arduino.

En primer lugar, comenzaremos con el joystick analógico, como ya explicamos consta de dos ejes, x e y en función de la dirección en la que tenga lugar el movimiento. Cada uno está asociado a un pin diferente. En nuestro montaje el eje x estará conectado al pin A0 de la placa mientras que el eje y hará lo mismo con el puerto A1.

Esta conexión se traduce en una activación de las salidas A0 y A1 del nodo DAQUINO_DIGIT_ANALOG, de forma que todo movimiento que realice el joystick en cualquiera de las direcciones se verá traducido en una salida tipo Float con valor entre 0 y 255.

Nótese la presencia de una fuente de Bit que será la que se encargue de alimentar y activar virtualmente al nodo DAQUINO_DIGIT_ANALOG.

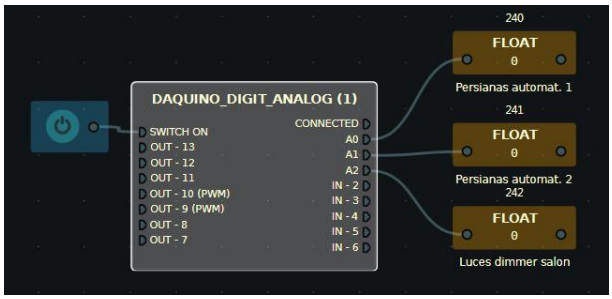


Figura 6-1 Conexión de nodos analógicos con Arduino

Para hacer más sencillo y compacto el diagrama de bloques emplearemos nodos aislados de tipo memorias a la salida del nodo DAQUINO_DIGIT_ANALOG, las cuales nos servirán como enlaces entre la comunicación del nodo Arduino y los diagramas de control de los elementos. Para la manipulación de las persianas hemos empleado las variables "Persianas automat 1" y "Persianas automat 2".

El tercer puerto restante del nodo DAQUINO_DIGIT_ANALOG que nos queda libre, A2, lo vamos a usar para conectar el diagrama de la luz dimmer del salón accionada mediante potenciómetro. La variable que vamos a usar para relacionar los nodos se va a llamar "Luces dimmer salón".

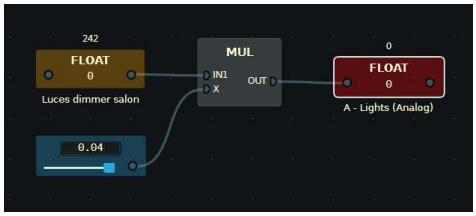


Figura 6-2 Diagrama de control de las luces del salón mediante potenciómetro

El circuito electrónico quedaría de la siguiente manera configurado:

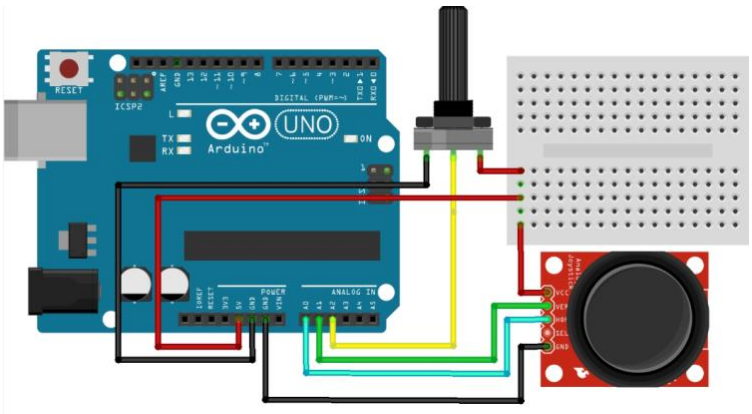


Figura 6-3 Configuración del montaje analógico

Con motivo de que en este apartado sólo vamos a emplear elementos analógicos se ha optado por modificar la programación de las luces del exterior de la casa empleando únicamente nodos analógicos.

Para ello simplemente tenemos que conocer el valor marcado por el sensor de brillo (Brightness Sensor) en las horas centrales de mayor exposición solar, marcando un valor de 8.6. Una vez conocido este valor lo introducimos en una fuente numérica y empleando un nodo de operación matemática tipo resta hacemos la función inversa regulando la luz durante todo momento.

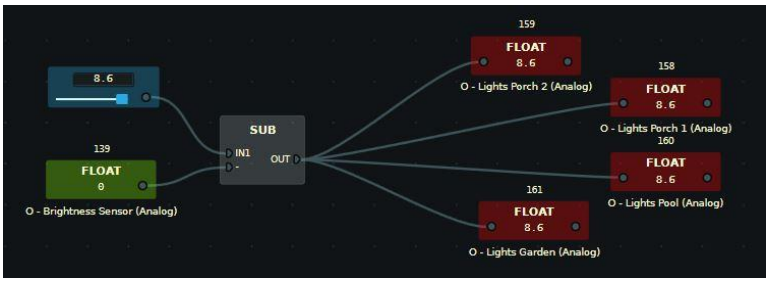
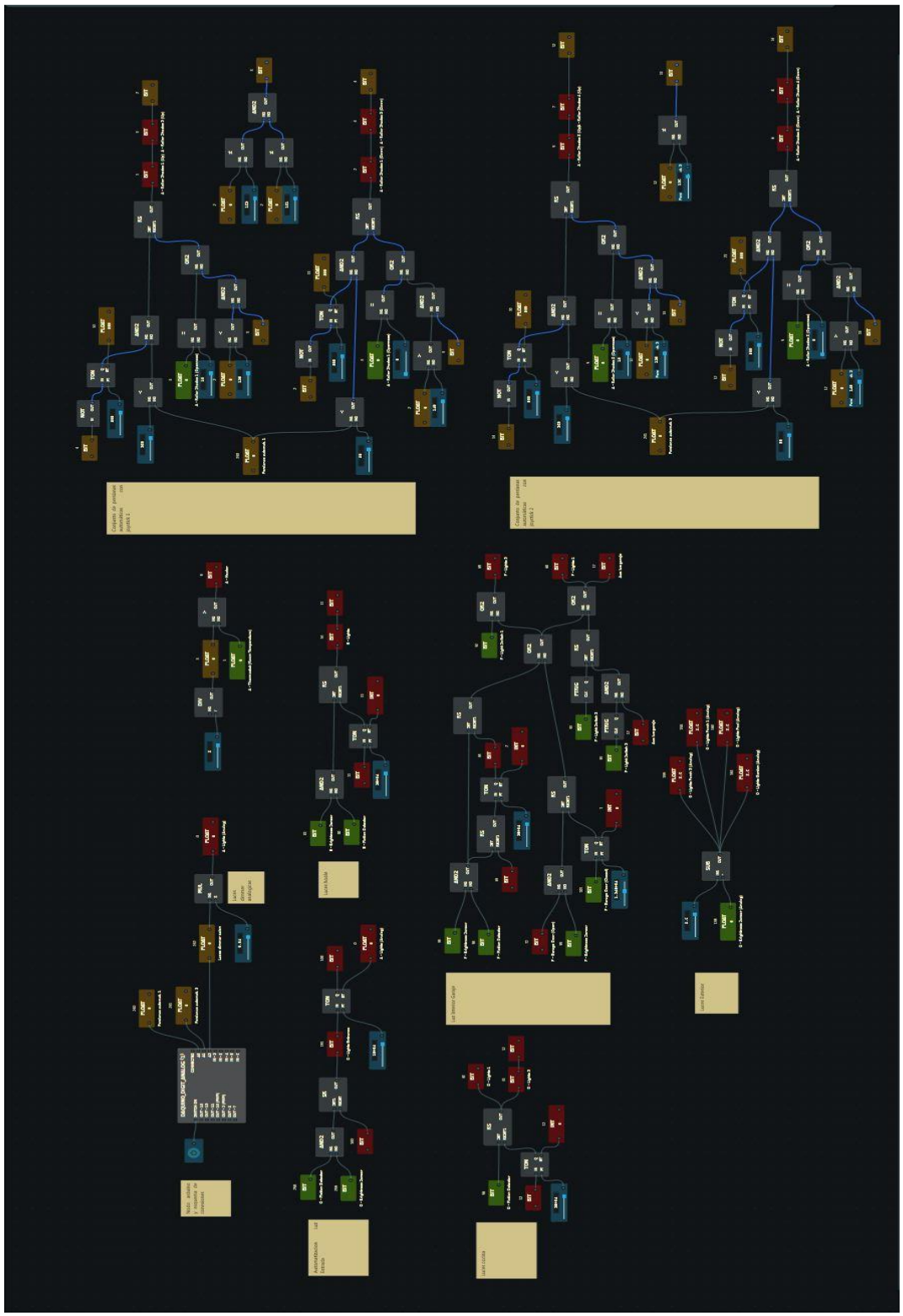


Figura 6-4 Accionamiento por nodos analógicos de las luces del exterior

Esquema del diagrama de bloques completo del fichero Arduino_analog.CONNECTIO

Figura 6-5 Contenido del fichero Arduino_analog.CONNECTIO



6.2 Configuración digital

Este es el apartado más complejo del proyecto, ya que tenemos que agrupar una gran cantidad de diagramas de bloques de cada uno de los elementos que se desean controlar, así como coordinar las direcciones de los nodos sin generar conflictos en una amplia lista de variables tanto tipo Bit como Float.

Por este motivo es muy recomendable ir pasando a un fichero nuevo en CONNECT I/O todos los diagramas e ir minuciosamente comparando las direcciones de las memorias auxiliares, ya que estas sobretodo son las que al tener valores arbitrarios dados por nosotros mismos nos pueden generar algún conflicto en la ejecución con nodos de tipo tags que ya traen de serie una dirección impuesta por el software HOME I/O.

En la configuración digital a diferencia de la analógica vamos a usar el nodo DAQUINO_DIGIT que sólo posee salidas digitales. De esta forma, los puertos digitales de la placa Arduino desde el 2 hasta el 7 estarán configurados como entradas para la placa y a su vez como salidas para el nodo DAQUINO_DIGIT.

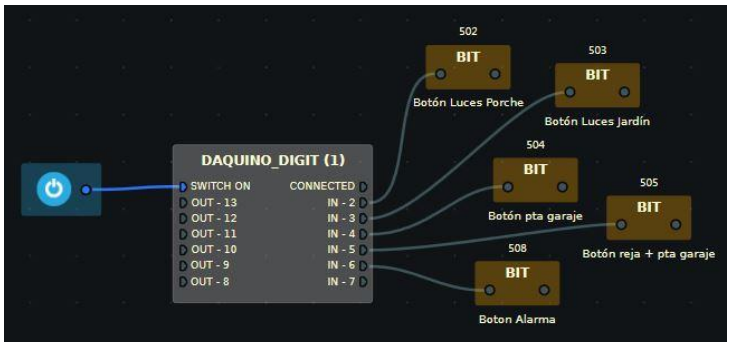


Figura 6-6 Conexión de nodos digitales con Arduino

Para el soporte real, debemos en primer lugar crear una serie de bloques de memoria de cada uno de los dispositivos ya programados siguiendo las instrucciones de las secciones anteriores. La misión de estas variables no es más que hacer de enlace entre el bloque DAQUINO_DIGIT y los diagramas de bloques digitales.

De esta forma simplificaremos y haremos mas legible el diagrama global. Entre otras cosas nos permitirá ordenar de forma correcta y a nuestro gusto todos los elementos.

Por tanto, en CONNECT I/O cada puerto estará conectado a un nodo de memoria que vendría a sustituir la función de los bloques asociados a cada uno de los botones del mando inalámbrico. La lista de variables de memoria quedaría así:

- Puerto digital 2: “Botón Luces Porche”
- Puerto digital 3: “Botón Luces jardín”
- Puerto digital 4: “Botón pta garaje”
- Puerto digital 5: “Botón reja + pta garaje”
- Puerto digital 6: “Botón Alarma”

El circuito electrónico quedaría de la siguiente manera configurado:

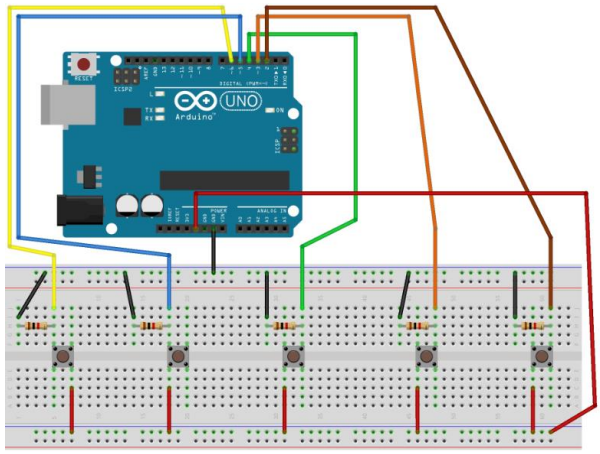


Figura 6-7 Configuración del montaje digital

Este sería el circuito electrónico o soporte real que hemos creado siguiendo todos los pasos hasta el momento:

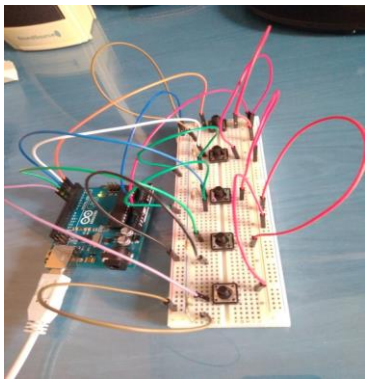


Figura 6-8 Imagen del soporte real diseñado

Siguiendo todas las pautas hasta ahora descritas, el siguiente paso sería ir trasladando cada uno de los diagramas al fichero final Arduino_digit.CONNECTIO. A continuación vamos a ir analizando de forma general cada uno de los cambios que habría que realizar de cara a su correcto funcionamiento:

- Comenzaremos con la programación de la alarma de seguridad de la casa inteligente. La gran diferencia con respecto a la activación de la alarma con el mando virtual reside en que tendremos que añadir una puerta lógica OR que enlace a la variable de memoria del soporte real “Botón alarma” y la entrada del mando inalámbrico “Remote button 6” cn el resto del diagrama. Es decir, este bloque OR y sus dos entradas vienen a sustituir al anterior bloque “Remote button 6”.

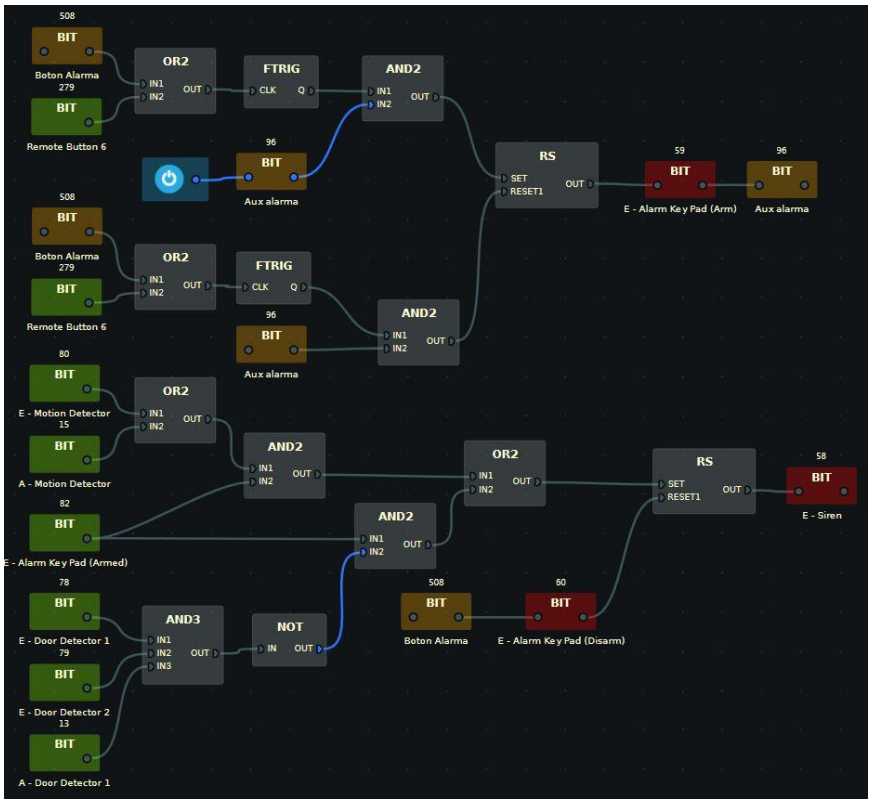


Figura 6-9 Modificación del diagrama de bloques del accionamiento de la alarma

Esta misma circunstancia tiene lugar también en la señal que se encarga de activar al Reset, con lo cual tendremos que aplicar la misma operación.

- Con la sincronización de la puerta del garaje nos ocurrirá exactamente lo mismo. Esta vez tendremos que ampliar el bloque OR por uno de tres entradas en lugar de dos. De forma que por una rama tengamos la activación por botones del mando inalámbrico, la segunda entrada pertenezca a los interruptores virtuales del garaje de la casa, y por último, a los botones del soporte real correspondientes a la apertura de la puerta del garaje y a la sincronización de la reja externa con las luces del jardín.

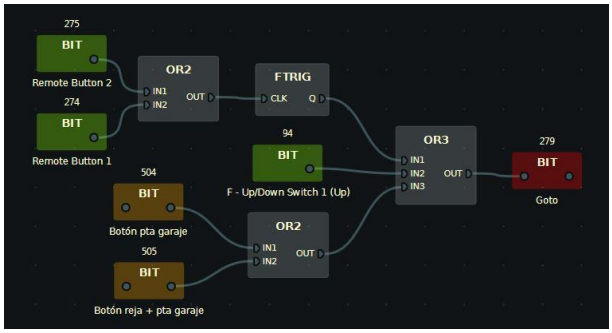


Figura 6-10 Modificación en la activación del nodo enlace de la puerta del garaje

- Las luces también tienen una configuración similar, ya que nos tendremos que centrar en dos aspectos a modificar: la parte correspondiente del diagrama al accionamiento de la luz y por otro lado la que se encarga de poner el estado HIGH al Reset y por tanto apagar la salida, las luces.

Como vemos la estrategia a seguir es muy parecida en todos los casos, siempre sustituyendo los

bloques iniciales asociados a los botones del mando virtual por la puerta lógica OR que nos permita accionarlos también por la señal del botón real facilitada por el Arduino, y en consecuencia por el nodo DAQUINO_DIGIT.

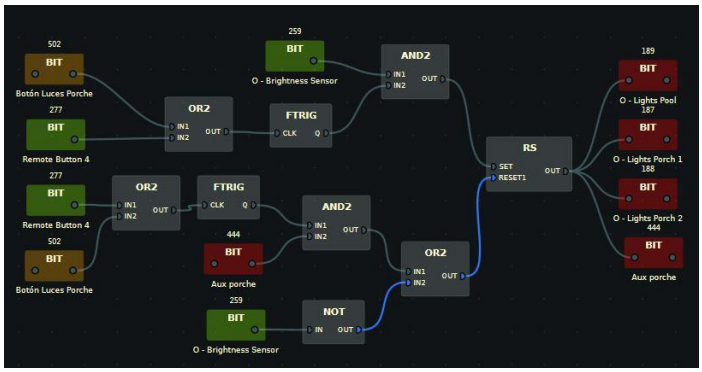
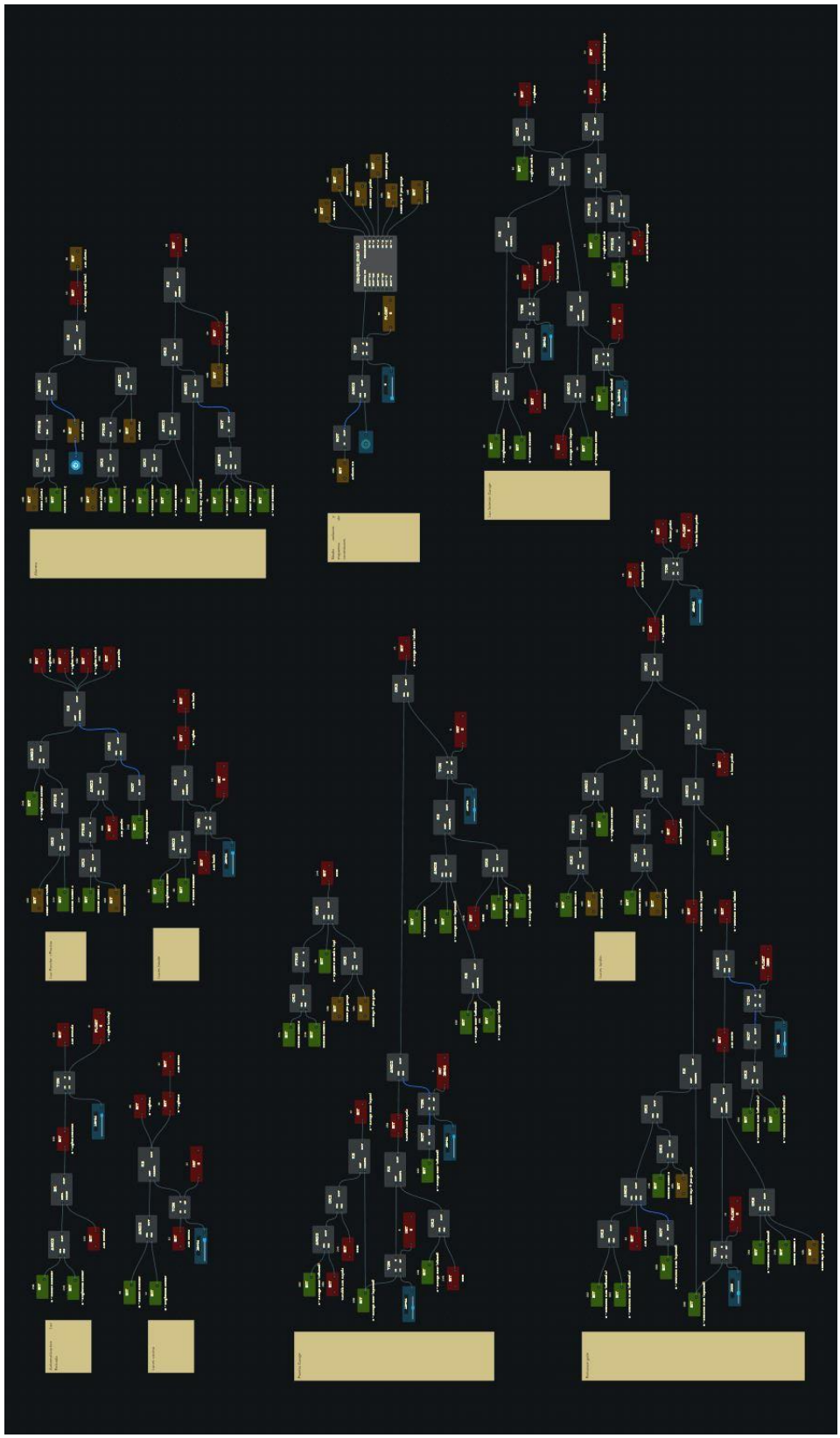


Figura 6-11 Modificación del diagrama de bloques del accionamiento de las luces del porche y jardín

Figura 6-12 Contenido del fichero Arduino_digit.CONNECTIO



6.3 Vídeos y consigna

- Canal de Adrián Barrera en la plataforma de Youtube:
  https://www.youtube.com/channel/UCMqfaKyi-m8OsQBKxXx5UUg

- Ficheros y videos para descargar en la plataforma de Google drive:
  -Archivos:
  https://drive.google.com/open?id=1AnwU1FE5eYQe7Yu91dXDVToXku980fMm

  -Vídeos
  https://drive.google.com/open?id=1o6NT44Ua-EJIDfxdkYmAfAfVhkNt9jcs

6.4 Resolución de problemas

Dado que durante el desarrollo de la aplicación hemos topado con varios problemas que si desconoces los motivos de su aparición pueden dar bastantes quebraderos de cabeza, hemos añadido esta sección a la memoria para ayudar al usuario a intentar resolverlos.

6.4.1 Puerto Serie COM256

Partimos de la base de que hemos seguido los pasos descritos en el apartado *4.3 Ajustar Puerto COM256* del capítulo *4 Comunicación entre Arduino y CONNECT I/O*, y hemos conseguido configurar correctamente dicho puerto serie para poder trabajar con él en el simulador.

Como ya deberíamos saber, el puerto serie de la placa Arduino sólo puede establecer comunicación con un elemento a la vez. Si lo estamos utilizando para subir un programa (o sketch), no lo podremos usar de manera simultanea para que trabaje con el simulador HOME I/O, o lo que es lo mismo, que en ese preciso instante esté establecida una comunicación con CONNECT I/O.

Esto último es de vital importancia puesto que en multitud de ocasiones si tenemos activado el nodo DAQUINO_DIGIT_ANALOG o su análogo DAQUINO_DIGIT mediante una fuente de Bit (Obviamente fuente en estado HIGH) y deseamos cargar cualquier programa al Arduino nos encontraremos con el siguiente mensaje:

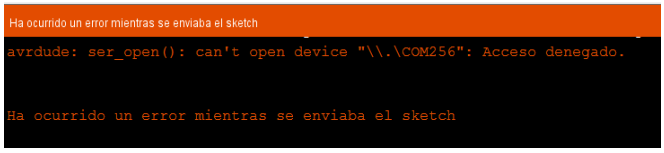


Figura 6-13 Error muy común mostrado por el IDE de Arduino

El acceso está denegado puesto que el puerto ya está siendo utilizado por otro dispositivo. La única manera de solucionarlo consiste es irnos a la interfaz de CONNECT I/O y apagar la fuente de Bit que alimenta al nodo DAQUINO_DIGIT_ANALOG o DAQUINO_DIGIT. Una vez ejecutado este paso se solucionarían el primer problema planteado del puerto COM256.

A medida que se va trabajando con la placa y los softwares, uno empieza a asimilar diversos mecanismos que indican que todo funciona correctamente o no. Uno de los detalles mas carácterísticos, es que cada vez que se active la fuente de Bit que alimenta al nodo de la placa, los leds TX y RX del puerto serie comenzarán a parpadear y se mantendrán.

Otro fallo muy común relacionado es cuando de buenas a primeras, la placa arduino no deja cargar ningún programa pero sin embargo aparentemente está todo correcto. Los leds de la placa siguen iluminándose pero sin embargo los leds TX y RX dejan de hacerlo.

Probablemente este es el problema mas indeseado puesto que su solución es la más enrevesada. Consiste en salvar todos los avanves de los programas HOME I/O, CONNECT I/O y cerrar también el software IDE de Arduino. Una vez hecho esto, debemos ir a panel de control y seguir los pasos ejecutados para configurar el puerto COM256 pero modificándolo por otro aleatorio que queramos. Guardamos la configuración para posteriormente volver a modificar la configuración y reestablecer el puerto COM256.

A partir de este momento quedan solucionados los problemas y todo vuelve a funcionar con normalidad para seguir trabajando.

6.4.2 La placa Arduino dejó de funcionar por completo

En algunas ocasiones cuando estas montando algún circuito en la protoboard con miras a llevar a cabo la comunicación entre algún dispositivo de HOME I/O y el soporte real, después de un engorroso montaje nos hemos encontrado con una situación un tanto irritante: nada funcionaba. Lo primero que se te viene a la mente es que ha podido deberse a un fallo al cargar el archivo parser en el arduino como nos ocurría en la sección anterior.

La parte más difícil es diagnosticar donde se encuentra el error para intentar subsanarlo lo más rápido posible y continuar con normalidad las pruebas del proyecto.

Finalmente, en ocasiones después de largo tiempo, descubríamos que el error se debía a que existía un cortocircuito en la portoboard. Ese era el motivo por el cual a la hora de volver a cargar el archivo parser a partir del software arduino nos apareciera el mensaje “No se puede cargar el programa”.

El mayor indicio para reconocer que nos encontramos en esta situación es cuando en la placa no se nos encienda ninguno de los leds que por norma general suelen estar iluminados.

6.4.3 Reconexión

Por último, probablemente por culpa del gran consumo de recursos, la pesadez de los archivos de CONNECT I/O debido a la gran cantidad de diagramas de programación, su archivo .dll y el parser del arduino generan algunos conflictos provocando que de forma aleatoria el sistema deje de funcionar.

La solución mas sencilla encontrada en primer momento consiste en estar activando constantemente de forma manual la fuente de Bit que enlaza con el nodo del Arduino cada vez que el nodo deja de funcionar (Esto lo podemos observar gracias a que su salida CONNECTED se apague de forma repentina).

Como resultaba poco profesional se ha trabajado en una solución que se encargue de hacer esta reconexión de forma automática. Para ello se ha planteado un bucle.

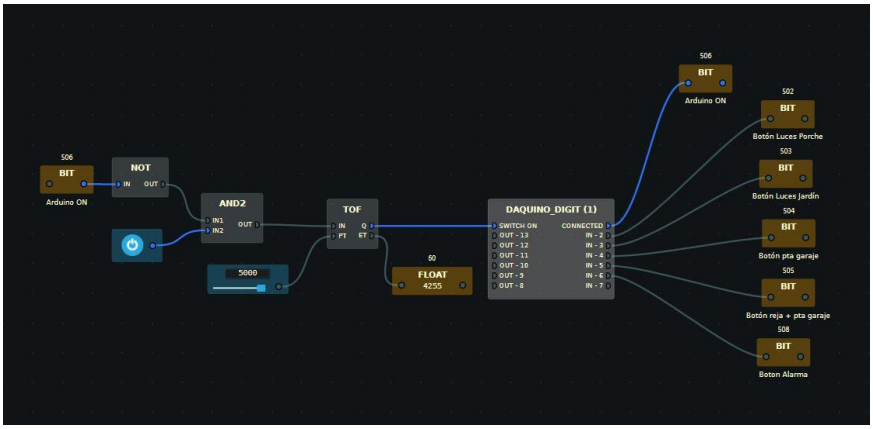


Figura 6-14 Bucle diseñado para la reconexión

Por la acción de un temporizador de retardo a la desconexión, TOF, cada 5 segundos se apaga y se enciende el bloque. Considerando que esta opción es mejor que estar manualmente haciéndolo cada poco tiempo, ya habríamos solucionado el problema a pesar de que cada 5 segundos haya un reinicio.

7 CONCLUSIÓN Y LÍNEAS DE DESARROLLO FUTURAS

Este capítulo de la memoria nos va a servir para plantear futuras líneas de trabajo o investigación así como hacer una breve valoración sobre lo aprendido durante el proyecto y qué nos ha aportado didácticamente.

7.1 Conclusión

Este proyecto desde un inicio se buscó enfocarlo en un marco muy práctico, como es la programación de los dispositivos domóticos que constituyen una vivienda.

Estas tecnologías, cada vez más presentes en nuestra vida cotiana, suponían una interesante línea de trabajo a partir de la cual construir o desarrollar un soporte real que nos permitiera controlar los distintos dispositivos programados previamente.

De ahí el gran interés, ya que a partir de las simulaciones tan reales de HOME I/O y gracias a su herramienta de programación CONNECT I/O se nos ofrece un mundo de posibilidades para investigar y mejorar el funcionamiento de algunos dispositivos. Ya no sólo desde el punto de vista de la programación, sino con miras a la eficiencia energética. El poder manipular a nuestro antojo las condiciones climatológicas o las horas de luz solar en función de nuestra posición en el globo nos permiten simular y ver como dichos elementos van a reaccionar ante tales condiciones.

Por supuesto dentro del desarrollo se ha tenido que hacer frente a multitud de situaciones o problemas generados por la programación de los nodos y comunicación entre sistemas. Ya sea por conflictos entre variables o porque la idea propuesta para alcanzar la meta no era la adecuada. En estos casos, se ha razonado y se ha revisado minuciosamente en busca del error o errores que nos hacían llegar a un callejón sin salida.

El mayor desafio ha sido adaptarse a un lenguaje de programación nuevo del que se tenía poca información, ponerlo en práctica y lograr conseguir buenos resultados a base de trabajo y horas de ingenio frente a la pantalla del ordenador.

7.2 Líneas de desarrollo futuras

Dentro de las posibles mejoras futuras destacaría varias ideas muy interesantes:

- Mejorar la comunicación entre Arduino y software HOME I/O.

  Por ejemplo, usando un recibidor de señales infrarrojas que se conecta a la placa Arduino. El módulo de sensor de infrarrojos detecta las señales infrarrojas y las retransmite a un sistema de control de Arduino compatible. La idea sería intentar simular de forma real el mando virtual que hemos estado empleando en la programación de las distintas acciones domóticas de la casa inteligente.

Figura 7-1 Módulo del recibidor de señales infrarrojas

Otra posibilidad sería la de aprovechar algunas apps prediseñadas por algunas compañías de Smartphones. Como es el caso de Xiaomi, con la app MIhome o MIremote que nos permiten utilizar nuestro Smartphone como mando infrarrojo.

También existiría la alternativa del control mediante la tecnología inalámbrica Bluetooth usando el módulo receptor para Arduino y estableciendo una comunicación de mayor alcance y velocidad con el software HOME I/O.

- Automatizar la casa al completo. Partiendo de la base ya construida, usar los distintos sensores de la casa inteligente para reconocer nuestra posición y habitación en la que nos encontremos. Establecer una relación posición-dispositivo para la recepción de órdenes.

- Investigar la línea de programación en CONNECT I/O con modularización en bloques.

- Código del archivo parser.ino empleado para realizar la pasarela entre el simulador HOME I/O y Arduino

```
//Librerias usadas
#include <stdlib.h>
#include <string.h>
#define NB_COMMANDS 7
#define MAX_ARGS 8
#define MAX_ARGS_LEN 40
//#define MAX_CMD_LEN 2048
#define MAX_CMD_LEN 40
//Identificadores de los comandos
enum Command_IDs {
  CReadAnalog = 0,
  CWriteAnalog = 1,
  CWriteDigital = 2,
  CReadDigital = 3,
  CReadTemperature = 4,
  CEcho = 5,
  CPrintConfirmation = 6
};
//Presentacion de los nombres de los comandos
char* Command_List[NB_COMMANDS] = {
  "ReadAnalog",
  "WriteAnalog",
  "WriteDigital",
  "ReadDigital",
  "ReadTemperature",
  "Echo",
  "PrintConfirmation"
};
//Comandos recibidos
int Command_ID;
char arg_list[MAX_ARGS][MAX_ARGS_LEN];
char incomingByte;
int index;
char command_line[MAX_CMD_LEN];
int iPrintConfirmation = -1;

//Imprime reconocimiento de un comando
void ack_command(char *message)
{
  if (iPrintConfirmation>0)
  {
    Serial.print("Command executed : ");
    Serial.print(Command_List[Command_ID]);
    Serial.print("->");
    Serial.print(message);
    Serial.println();
  }
}

//Funcion que ejecuta los comandos
void exec_command()
{
```

```
  char user_feedback[512];
    switch (Command_ID)
    {
      case CReadAnalog:
        Serial.println(analogRead(atoi(arg_list[1])));
       ack_command("done");
        break;
      case CWriteAnalog:
        analogWrite(atoi(arg_list[1]), atoi(arg_list[2]));
       sprintf(user_feedback, "done : Inputs %i,%i", atoi(arg_list[1]),
atoi(arg_list[2]));
        ack_command(user_feedback);
        break;
      case CReadDigital:
        Serial.println(digitalRead(atoi(arg_list[1])));
        ack_command("done");
        break;
      case CWriteDigital:
        digitalWrite(atoi(arg_list[1]), atoi(arg_list[2]));
        sprintf(user_feedback, "done : Inputs %i,%i", atoi(arg_list[1]),
atoi(arg_list[2]));
        ack_command(user_feedback);
        break;
      case CEcho:
        Serial.println(arg_list[1]);
       ack_command("done");
        break;
      case CPrintConfirmation:
        iPrintConfirmation = atoi(arg_list[1]);
       ack_command("done");
        break;
      default :
        Serial.print("Unknown Command");
        Serial.println(command_line);
    }
}
//Esta function analiza los commandos con el objetivo de identificar el
commando y sus argumentos
void parse_command()
{
  char * pch;
  int ii = 0;
  //Identifica los atributos dle comando
  pch = strtok (command_line,"|/Ã,Â¤");
  while (pch != NULL)
  {
    sprintf(arg_list[ii],"%s", pch);//Escribe en la la table de argumentos
    pch = strtok (NULL, "|/Ã,Â¤");
    ii++;
  }
  //Identify the command ID
  for (ii=0; ii<NB_COMMANDS; ii++)
  {
    if (strcmp(Command_List[ii], arg_list[0]) == 0)
    {
```

```
      Command_ID = ii;
      break;
    }
  }
}
//Basic setup. Adapta el contenido de acuerdo a las entradas y salidas void
setup() {
    for(int i = 0; i < 6; i++)
    {
     pinMode((i+2), INPUT); // pin #1 et #2 indisponible, communication série
rx tx
     pinMode((13 - i), OUTPUT);
    }
    Serial.begin(9600);
}
//Esperar para los commandos de entrada
void loop() {
  if (Serial.available()) {
    incomingByte = Serial.read();
    switch (incomingByte)
    {
      case '<': //Empezar adquisicion de un nuevo comando
        index = 0;
        memset(command_line, 0, MAX_CMD_LEN - 1);//Set command to 0
        break;
      case '>': // Terminar adquisicion del comando actual
        command_line[index+1] = 0;
        parse_command();
        exec_command();
        break;
      default : // adquisicion de comando
        command_line[index++] = incomingByte;
    }
  }
}
```

REFERENCIAS

[1] «https://www.arkiplus.com/historia-de-la-domotica,» [En línea].

[2] «https://aprendiendoarduino.wordpress.com/2016/09/25/que-es-arduino/,» [En línea].

[3] «https://aprendiendoarduino.wordpress.com/tag/fritzing/,» [En línea].

[4] «http://www.casasdigitales.com/curiosidades-domoticas/,» [En línea].

[5] «https://es.wikipedia.org/wiki/Dom%C3%B3tica,» [En línea].

[6] «https://es.wikipedia.org/wiki/Zigbee,» [En línea].

[7] «https://forum.arduino.cc/,» [En línea].

[8] «https://realgames.co/docs/connectio/,» [En línea].

[9] «https://realgames.co/docs/homeio/en/,» [En línea].

[10] «https://www.cervi.es/ES/3-productos/19--cables-de-sistemas-bus/294-cable-eibknx.html,» [En línea].