UNIVERSIDAD DE OVIEDO

MÁSTER EN INGENIERÍA WEB

CENTRO INTERNACIONAL DE POSTGRADO

TRABAJO FIN DE MÁSTER

Desarrollo e implementación de integradores de plataforma utilizando Fi-Ware

AUTOR: Ricardo Diez Villamañán
DIRECTORES: Juan Manuel Cueva Lovelle, Cristian González García

Agradecimientos

En primer lugar Juan Manuel Cueva Lovelle por la posibilidad de la tutorización, a Cristian por su labor impagable de co-tutorización, mentor, motivador, consejero y amigo que cualquier ponente necesita.

A mis padres por el apoyo logístico y moral desde el primer momento.

A mis compañeros de laboratorio Daniel Meana, Claudio, Bernardo por su ayuda desinteresada y sobre todo a Daniel Fernández por la multitud de horas compartidas en el último piso del edificio de ciencias.

Y como no, a Marina por no dejar de ayudar y de apoyar en todas las dificultades que fueron aconteciendo dentro y fuera del proyecto, siendo el combustible que no permitió que esta maquinaria se detuviera.

Resumen

El objetivo del proyecto es crear un prototipo para un sistema de control de temperatura en un lugar determinada utilizando Internet de las Cosas para interconectar objetos. El prototipo será dotado de Inteligencia Artificial a través del uso de lógica difusa para su automatización.

Por un lado se van a consumir datos de algunas de las plataformas Internet de las Cosas que se encuentran abiertas a usuarios ofreciendo información de multitud de sensores localizados en distintos lugares.

Por otro lado se van a consumir datos de varios sensores instalados e implementados manualmente en otros tipos de dispositivos, como pueden ser microordenadores o microcontroladores. Estos datos se subirán a alguna de las redes Internet de las Cosas con el propósito de que se puedan consumir posteriormente por cualquier usuario.

De esta manera, se pretende realizar un prototipo que sea capaz de manejar automáticamente una estufa y un aire acondicionado de un determinado sitio, según unas reglas definidas utilizando lógica difusa, para así estabilizar la temperatura de ese lugar en función de la temperatura externa e interna.

Palabras Clave

Internet de las cosas, Raspberry Pi, Lógica Difusa.

Abstract

The target of the prototype is to create a temperature control system for a specific place using the Internet of Things to interconnect objects. The prototype will be equipped with artificial intelligence through the use of fuzzy logic for create an automated system.

On one side it is going to consume data from some of the Internet of Things platforms that are open to users, providing information and multitude of sensors located in different places.

On the other side it is going to consume data from multiple sensors implemented and manually installed on other types of devices, such as microcomputers or microcontrollers. These data will be uploaded to any platform of the Internet of Things so that it possible later be consumed by any user.

In this way, we intend to make a prototype that is able to automatically handle a stove and air conditioning for a specific place, according to rules defined using fuzzy logic in order to stabilize the temperature of the place depending on external and internal temperature.

Keywords

Internet of things, Raspberry Pi, Fuzzy Logic.

Índice General

Índice de Figuras

Capítulo 1. Memoria del Proyecto

1.1 Resumen de la Motivación, Objetivos y Alcance del Proyecto

La principal motivación para la realización de este proyecto es la creación de un prototipo dotado de inteligencia artificial que, a través de unas reglas establecidas aplicadas con lógica difusa, logre una automatización concreta.

Por otro lado, el gran aumento en los últimos tiempos del uso de *Internet de las Cosas (IoT*), de los *Smart Objects*, *Smart Cities*, así como de la aparición de sensores unidos a la evolución de los microprocesadores como *Raspberry* o microcontroladores como *Arduino*, hacen que el trabajo con esta tecnología sea muy interesante.

Internet de las cosas es utilizado para medir multitud de parámetros, tanto externos como internos, abarcando una gran variedad de sectores que, de forma automática, son procesados y permiten tomar decisiones en tiempo real, siempre intentando facilitar o remediar necesidades para el ser humano y creando fuentes de información con miles de datos listos para utilizarse.

Muchos de los datos que existen en relación a *Internet de las cosas*, son sensores que captan información referida a condiciones ambientales tales como temperatura, humedad, luminosidad y presión, lo que ayuda al desarrollo de proyectos que se interactúen con estos datos.

En la actualidad existen varias plataformas *IoT* que permiten tanto publicar datos de sensores creados por uno mismo como visualizar y consumir los que otros usuarios ofrecen públicamente.

Por todo lo mencionado, el objetivo previo del trabajo es la creación de un prototipo que englobe los puntos anteriores, logrando una automatización de un sistema que controle las condiciones ambientales de un determinado lugar, consiguiéndose a su vez un ahorro de la energía.

Para ello se han desarrollado e instalado sensores de temperatura localizados en un micro ordenador como *Raspberry Pi* y en un micro controlador como *Arduino*. Externamente, se han utilizado plataformas *IoT* para consumir datos de temperatura y de humedad localizados en el exterior.

Estos datos serán posteriormente tratados con lógica difusa para poder otorgar al prototipo de cierta inteligencia y conseguir automatizar su funcionamiento.

En un principio se comenzó a investigar y utilizar la plataforma Fi-Ware junto a algunas de las API´s que proporcionaba, usando los servicios on-line que tiene para el desarrollo de

aplicaciones como por ejemplo el laboratorio de desarrollo para experimentar con sus General Enblers, https://account.lab.fiware.org/ .

Pero a medida que pasaba el tiempo esta plataforma conjunto con la información que existe en la web no me ofrecía la posibilidad ni la seguridad de poder experimentar el tipo de prototipo que quería realizar a corto plazo, por lo que opte por cambiar de tecnología para el desarrollo del mismo.

Con esta decisión no se quiere insinuar que el uso de FI-Ware no pueda proporcionar una solución usando algunas de sus API´s, sino que en este caso, al depender de una fecha de entrega final, no disponía de todo el tiempo que requiere la investigación de una nueva plataforma como es Fi-Ware.

1.2 Resumen de Todos los Aspectos

1.2.1 Introducción

El Internet de las cosas apunta a ser una de las grandes tendencias a partir del 2015, ya que se trata de una tecnología que es capaz de conectar todo tipo de dispositivos a la red, comunicarse y realizar acciones entre sí.

Esta tendencia busca la interconexión de todo tipo de dispositivos y que los sensores localizados en todos estos dispositivos no solo permitan la comunicación sino que dependiendo de estos datos recogidos envíen instrucciones para realizar acciones sencillas que en conjunto sean capaces de aportar valor y automatizar procesos.

Por ello se decide a crear un prototipo que recoja de manera específica y entrando en detalles de algunos aspectos que rodean el amplio entorno del Internet de las cosas.

1.2.2 Análisis y Diseño

El proyecto que se desarrollará será, como se dijo, será un prototipo de control de temperatura que, mediante el consumo de datos de sensores situados en distintos lugares y procesados a través de lógica difusa, se le pueda otorgar cierta inteligencia artificial para automatizar el proceso de funcionamiento.

El prototipo consumirá datos de:

- Sensores físicos
- Plataformas de Internet de las cosas.
- Datos simulados

El prototipo simulará el accionado de controladores de temperatura como pueden ser aires acondicionados o termostatos para lograr mantener unas condiciones óptimas identificadas con anterioridad en el lugar que se quiera utilizar.

1.2.3 Planificación y Presupuesto

Se ha planificado que el proyecto sea realizable en algo más de 4 meses y medio contando con toda la investigación anterior realizada con respecto a tecnologías a usar para el desarrollo del mismo y un presupuesto aproximado a 44.415,€

1.2.4 Implementación

Para la implementación del sistema primero se adquirió un microcontolador Arduino, y un microordenador Raspberry que será desde donde se desarrollará y ejecutará el prototipo en un inicio.

La aplicación fue desarrollada totalmente en un entorno de desarrollo Java a excepción de algún módulo para obtener datos del Arduino que se realizó en C.

Capítulo 2. Introducción

2.1 Justificación del Proyecto

El desarrollo de este proyecto viene dado por dos aspectos:

- El desarrollo de aplicaciones con *Internet de las cosas* ha aumentado notablemente. Cada vez son más los objetos que están conectados a Internet e interconectados entre sí con el objetivo de realizar acciones sin necesidad de la acción humana y convertir cualquier elemento en inteligente (*Smart Cities, Smart Objects*).
- Unido al punto anterior de conseguir elementos dotados con la mayor inteligencia artificial posible y lograr automatizaciones de sistemas, se ha añadido el uso de la lógica difusa donde se definen reglas para satisfacer, específicamente, las peticiones que se requieren.

Ante estas dos premisas se eligió como ejemplo de desarrollo el control de la temperatura en un determinado lugar, esto es, simular el funcionamiento de una calefacción y del aire acondicionado de una sala para conseguir que ésta sea inteligente y mantenga unas condiciones ambientales adecuadas a la vez que ahorre en energía sin necesidad de estar pendiente de ello.

Se van a tener en cuenta las condiciones externas del lugar que queremos controlar para así obtener la sensación térmica exterior. Puesto que no se ha podido contar con sensores continuos externos, se han separado en dos las vías para conseguir la temperatura y la humedad con las que se va a calcular la sensación térmica.

Para el cálculo de la humedad se van a consumir los datos desde una plataforma *IoT* llamada *ThingSpeak*, que nos ofrece una multitud de canales abiertos donde consultar datos. Por otro lado, para obtener la temperatura exterior, se van a conseguir desde un micro controlador *Arduino* que contiene un circuito con un sensor de temperatura que ya estará previamente programado para servir la temperatura que simulará la temperatura exterior.

Ambas condiciones externas servirán para que, a través la lógica difusa, así como de un primer bloque de reglas definidas, se pueda calcular la sensación térmica exterior que nos servirá como entrada para el siguiente paso.

Con un segundo bloque de normas también establecidas con Lógica difusa, se usará la sensación térmica exterior anteriormente calculada y la temperatura local que será obtenida por un sensor instalado en un micro procesador *Raspberry*.

Finalmente, obtendremos un valor cuya salida se representará con un conjunto de Led´s instalados en la *Raspberry*, ya que no contamos con la posibilidad de interactuar con un aparato de control de temperatura como una calefacción o aire acondicionado.

Se mostrarán 5 Led's: dos de ellos rojos, referidos a dos niveles distintos de potencia de la calefacción, dos Led's azules que, paralelamente, representarán otros dos niveles distintos de

potencia de un aire acondicionado y, por último, un LED tricolor que mostrará un estado óptimo de condiciones ambientales dentro de la sala, teniendo como objetivo poder alcanzar ese estado o bien mantenerlo.

2.2 Objetivos del Proyecto

El primer objetivo es empezar a trabajar con *Internet de las cosas* y con los elementos que lo rodean, como son los *Smart Objects*, sensores, microcontroladores o microprocesadores.

Con ello, el siguiente objetivo y el más importante, es realizar un prototipo completo que satisfaga unas pautas marcadas que se desglosan a continuación:

- Consumir datos de plataformas de *Internet de las cosas* abiertas a usuarios.
- Consumir datos de sensores instalados en micro controladores como *Arduino* que están localizados en redes o entornos exteriores.
- Trabajar con los actuadores y sensores de una *Raspberry*, así como con los datos que se muestran.
- Establecer reglas de lógica difusa para conseguir un mejor rendimiento en la solución del problema planteado.
- Mantener una temperatura óptima dentro de la sala durante el mayor tiempo posible.
- Ahorrar en el consumo de energía lo cual conlleva a un ahorro económico.

2.3 Estudio de la Situación Actual

2.3.1 Evaluación de Alternativas

2.3.1.1 *Proyectos*

A continuación se muestran algunos de los proyectos sobre los que se investigó para comprobar el funcionamiento de los mismos, ya que son sistemas que ya están en el mercado y se puede comprobar y ver su funcionamiento en aplicaciones reales.

Por otro lado se mostrarán varios trabajos donde se exponen la utilización de lógica difusa para el control de temperatura pero solo en el ámbito teórico.

2.3.1.1.1 Loxone

2.3.1.1.1.1 Descripción

Ilustración 1 Logo de Loxone

Loxone es un sistema domótico muy potente de nueva creación que abarca varios ámbitos, como el control de persianas, luces, equipos de música o aparatos de climatización.

En este caso se evalúa únicamente el sistema que engloba el control de temperatura de una casa (calefacción y aire acondicionado) aunque existen dependencias con algunos de los otros aparatos domotizados, como pueden ser las persianas.

El objetivo final de este sistema es, a través de la automatización, obtener un ahorro económico y un control simple desde aplicaciones o Interfaz Web que facilite el acceso remoto.

2.3.1.1.1.2 Ventajas

- Acceso al control de temperatura vía Interfaz Web o aplicaciones.
- Sistema domótico con más funcionalidades como apertura de persianas que van a influir en el control de la temperatura.
- Uso de geolocalización para el control del funcionamiento dependiendo de si el usuario se encuentra en casa o no.
- Registro de temperaturas almacenado para estudiar tendencias y funcionamientos correctos para lograr ahorros potenciales.
- Configuración de escenas predeterminadas como situaciones especiales, con características particulares según convengan.
- Núcleo de soluciones en el *MiniServer* donde se pueden conectar varios tipos de dispositivos aprovechando sus entradas y salidas digitales y analógicas.

- Integración de los sistemas, de frío – calor, entre otros, en el *MiniSever* que controla todas las tareas evitando que, si existen cambios de temperatura inesperados, se activen los dos sistemas a la vez.

2.3.1.1.1.3 Inconvenientes

Aunque se trata de sistemas que tienen una gran cantidad de ventajas, también existen algunos inconvenientes o diferencias que fomentan las bondades de mi prototipo. Éstas son:

- Alto precio de compra.
- Complicada instalación del sistema de calefacción, teniendo en cuenta que para cada sala se requieren nuevos termómetros e instalaciones.
- Bajo nivel de prestaciones si no se usa en una casa domotizada.
- Alto coste de la instalación domótica de elementos como pueden ser persianas, puertas y detectores de presencia.
- Solo utiliza datos que se encuentran dentro de sus redes, no logra obtener datos de fuera de la red local del lugar.

2.3.1.1.2 tado°

2.3.1.1.2.1 Descripción

Ilustración 2 Logo tado°

"Tado°" es un termostato inteligente diseñado por una empresa alemana. Se trata de un termostato como cualquier otro que se va a encontrar en el lugar que se quiere controlar, iniciando y parando la calefacción según sea necesario; la diferencia es que lo hace de un modo inteligente gracias a que tiene más datos que un sensor normal. Puede ser controlado desde Interfaz Web o desde un Smartphone y utiliza la red Wi-fi de su casa para estar constantemente conectado a su móvil.

"Tado°" también posee un sistema de aire acondicionado con un funcionamiento similar al termostato anterior, pero siendo dos elementos totalmente distintos. En este caso, *"Tado°"* reemplaza su anterior control remoto del aire acondicionado por un dispositivo pequeño que se colocará cerca de su aparato y que estará conectado a la red Wi-fi de su casa para poder obtener la información y los datos con los que trabaja.

2.3.1.1.2.2 Ventajas

Al hablar de *"Tado°"*, nos referiremos al conjunto de ambos dispositivos ya que las características con las que trabaja son similares, tanto para el uso del termostato como del aire acondicionado.

- Uso de geolocalización para programar el funcionamiento de los aparatos según se encuentres o no en casa.
- Control remoto desde cualquier Interfaz Web o aplicaciones para dispositivos móviles.
- Ahorro de dinero.
- Instalación sencilla:

  - Calefacción: reemplazamiento del termostato anterior por uno nuevo con conexión a internet.
  - Aire acondicionado: reemplazamiento del control remoto por un sistema fijo cerca del aparato.
  - Conexión a Internet a través de la red Wi-Fi de su casa.
- Informes detallados del uso de los aparatos para comprobar el ahorro.
- Utiliza el pronóstico del tiempo para configurar y mantener la temperatura ambiente.

2.3.1.1.2.3 Inconvenientes

- Alto precio de compra, puesto que son dos productos diferentes para albergar el control total de la temperatura.
- No usa ninguna lógica para considerarse inteligente, ya que solo trabaja con datos de configuración estipulados por el usuario.
- Realiza variaciones de temperatura respecto a las previsiones de temperatura que, en ocasiones, pueden ser erróneas.
- Solo usa termómetros de dentro de la casa integrados en el circuito habitual de calefacción.

2.3.1.2 *Trabajos teóricos.*

2.3.1.2.1 Descripción

Existe una gran variedad de trabajos teóricos referidos a la creación de sistemas de control de temperatura a través de lógica difusa. Varios de los que se han evaluado con mayor profundidad tienen características comunes por lo que se han englobado en este apartado.

Se trata de trabajos que tienen como fin controlar la temperatura utilizando la lógica difusa como herramienta de toma de decisiones.

En todos los casos definen las variables de entrada y de salida del sistema a la vez crean la identificación de términos lingüísticos a cada variable. Construyen reglas difusas para cada asociación difusa y por último seleccionan la implementación y parámetro de los operadores difusos que utilizan.

En todos los casos se definen las variables de entrada y de salida asociadas a términos lingüísticos determinados. Además, crean reglas difusas para cada asociación difusa y, por último, seleccionan la implementación y los parámetros de los operadores difusos que utilizan.

2.3.1.2.2 Ventajas

- Para la demostración de trabajo con lógica difusa, se toma como ejemplo situaciones relacionadas con la temperatura puesto que es un claro ejemplo donde la lógica difusa resulta efectiva.
- Se mezclan distintos tintos de variables referidos a condiciones ambientales, lo que ayuda a definir el sistema.

- Las salidas que muestran exponen siempre un control de temperatura, ya sea representado con aperturas de válvulas de aire o de cualquier variador de temperatura.

2.3.1.2.3 Inconvenientes

- En cada caso sólo se usan dos variables de entrada como máximo.
- No se realiza ningún prototipo físico en el que se pueda mostrar el funcionamiento.
- Los datos recogidos siempre son locales, no se consumen a través de ninguna plataforma de *Internet de las cosas*.

2.3.1.3 *Vitruvius*

2.3.1.3.1 Descripción

Hasta ahora solo se ha hecho referencia a la aplicación de la lógica difusa en ámbitos relacionados con condiciones ambientales, pero la lógica difusa abarca una gran multitud de ámbitos como puede ser el siguiente ejemplo descrito en el siguiente artículo científico referenciado [13].

Vitruvius es una plataforma en la que se permite a los usuarios generar aplicaciones en tiempo real usando sensores instalados en los vehículos de carretera. La aplicación Web consiste en un editor de aplicaciones que permite a los usuarios diseñar aplicaciones basados en sensores.

Los usuarios pueden elegir entre un amplio rango de sensores, como velocidad, refrigeración del motor, temperatura, posición del acelerador, presión de aire…

En este caso la lógica difusa se utilizará para reducir la frecuencia con que los datos de los sensores del vehículo son enviados al servidor de datos central. En la versión anterior del sistema los datos se enviaban cada segundo al servidor a través del servicio web.

A través de la definición de un conjunto de reglas difusas se configura la habilidad de posponer el envío de alguno de los datos y unificar las peticiones del servicio web cuando las variaciones de los envíos de datos no sean necesariamente significantes para la aplicación.

2.3.1.3.2 Ventajas

- Menos cantidad de datos enviados.
- Mejor calidad de los datos enviados.
- Mejor rendimiento de los análisis realizados.

2.3.1.3.3 Inconvenientes

- Se existe error en algunos de los sensores, mayor dificultad a la hora de percatarse de dicho error.

2.3.1.4 *Hardware*

2.3.1.4.1 Microordenador VIA APC

2.3.1.4.1.1 Descripción

VIA APC es un microordenador basado en una arquitectura ARM y Android como sistema operativo, viene en formato llamado Neo-ITX que solamente ocupa 17cm x 8.5 cm y es posible adaptarlo a pequeñas cajas Mini-ITX. En este caso la memoria ya viene integrada con 2GB de NAND Flash, aunque se pude ampliar con un una tarjeta micro SD. Posee 4 puertos USB 2.0, conectores de audio y dos salidas de video. Se trata de una de los múltiples competidores que el microordenador Raspberry Pi tiene.

2.3.1.4.1.2 Ventajas

- Tamaño más pequeño.
- Mejores prestaciones.

2.3.1.4.1.3 Inconvenientes

- Viene con un sistema operativo Android ya instalado.
- Más caro que Raspberry Pi.
- Problemas al instalar sistema operativo Linux
- Comunidad de usuarios minoritaria lo que implica menos recursos en la red.

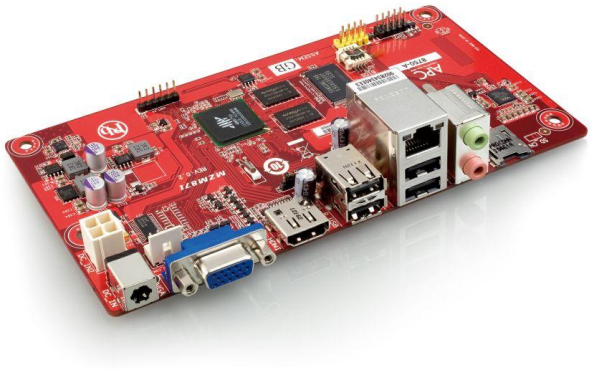


Ilustración 3 VIA APC

2.3.1.4.2 Intel Edison

2.3.1.4.2.1 Descripción

Intel Edison puede definirse como un PC completo del tamaño de una tarjeta SD estándar, unos 32x24x2, 1 milímetros, un mini PC enfocado a una cota de mercado muy concreta, desde Intel, se quiere lograr que *Edison* sea el centro de toda la informática "*para llevar puesta*".

Se trata de un mini PC que contiene una CPU de doble núcleo y un microcontolador de un solo núcleo. Wi-fi integrada, Bluetooth, memoria, almacenamiento y 40 interfaces GPIO

multiplexadas con opciones de placa de expansión para una total flexibilidad y diseño de proyectos.

2.3.1.4.2.2 Ventajas

- Reducido tamaño para incorporar en wearables y objetos (ropa, juguetes, mesas, sillas…).
- Puente entre sensores y otros dispositivos
- Compatibilidad alta con estándares como Wi-Fi o Bluetooth
- Soporte para C, C++, Python, Node.js/Javascript para desarrollar aplicaciones.

2.3.1.4.2.3 Inconvenientes

- Precio más elevado que Raspberry PI
- No tienen entrada de video
- No tiene entradas USB
- Velocidad lenta del procesador
- Las conexiones de entrada y salida son imposible si no se usa una placa extra.

Ilustración 4 Intel Edison

2.3.1.5 *Plataforma*

2.3.1.5.1 Fi-Ware

2.3.1.5.1.1 Descripción

Ilustración 5 Logo Fi-Ware

Fi-Ware es una plataforma que represente una opción abierta para el desarrollo y el despliegue global de aplicaciones en el Internet del futuro. Esta plataforma está fuertemente apoyada la Comisión Europea (CE) y las principales empresas TIC Europeas

están invirtiendo fuertemente en ello.

2.3.1.5.1.2 Ventajas

- *Fi-Ware* proporciona un conjunto de APIs, abiertas y completamente libres de royalties, de manera que ayude al desarrollo rápido de aplicaciones en numerosos sectores e implantación de nuevos servicios, facilitando la reutilización e introduciendo estándares.
- Está formada por *Generals Enablers* que son uno conjunto de elementos básicos reutilizables con los que se pueden construir nuevos servicios y aplicaciones en campos tan distintos como Smart Cities, internet de las cosas, e-salud, e-educación, Big Data…
- Paralelo a esta gran inversión de dinero en el fomento de desarrollo de aplicaciones bajo esta tecnología se han creado programas de aceleración para nuevos proyectos.

2.3.1.5.1.3 Inconvenientes

- Después de varios meses de investigación y de poco desarrollo en esta plataforma con algunos de los General Enablers que se ofrecían, no se consiguió ningún resultado que pudiera mostrar habiendo invertido multitud de tiempo en ello.
- Las principales causas de desistir de seguir utilizando esta tecnología fueron causadas por la poca estabilidad que las máquinas virtuales ofrecían.
- Estas máquinas que caían durante periodos de tiempo sin ningún aviso tenían instalados algunos de los General Enablers con los que se quería trabajar.
- También existía el problema de que no se podían instanciar varios GE en una misma máquina, ya que habían sido desarrollados en distintos sistemas operativos lo que hacía que el trabajar con varios GE significara trabajar con destinas máquinas con distintos sistemas operativos.
- Gran dificultad a la hora de interactuar con los General Enablers entre sí.

Por estos y otros inconvenientes después de algún tiempo decidí cambiar en el uso de esta plataforma ya que a medida que pasaba el tiempo no obtenía ningún resultado claro y el tiempo se me acababa para completar el desarrollo de mi proyecto que estaba marcado con una fecha final de entrega.

Posiblemente con suficiente tiempo dedicado podría haber llegado a conseguir el objetivo final usando la plataforma Fi-Ware, pero en este caso, dada mi situación no fue posible el uso de tanto tiempo.

Capítulo 3. Aspectos Teóricos

3.1 Internet of Things

Actualmente se puede apreciar un notable crecimiento del uso de *Internet de las Cosas*, es decir, cada vez son más los *Smart Objetcs* que se conectan a Internet interactuando entre sí, intercambiándose datos y realizando determinadas acciones en función de la información requerida.

El concepto de *Internet of Things* se refiere a la interconexión digital de objetos heterogéneos conectados a Internet. Surge debido a la necesidad de las cadenas de suministro y de identificar objetos, personas y animales mediante el uso de etiquetas inteligentes RFID. Éstas son colocadas en un objeto pudiendo así localizarlo para determinadas tareas.

Según Tan (2010) se necesitan tres pasos para la existencia de *Internet of Things*: inteligencia integrada, conectividad e interacción. Por tanto, se puede decir que la base está compuesta por sensores y tarjetas RFID [1], pero no son los únicos.

En resumen, *Internet de las cosas* está basado en sensores, redes de comunicaciones y en una inteligencia que maneja todo el proceso con los datos generados. Los sensores, que deben de tener un bajo consumo y coste, así como un tamaño reducido y gran flexibilidad para su uso en todo tipo de circunstancias, van a servir para la recogida de casi cualquier tipo de información o dato. Estos datos precisan de una infraestructura inteligente que sea capaz de conectar los distintos objetos y que puedan ser otorgados de inteligencia para interactuar entre sí.[2]

Un aspecto importante del *IoT* que servirá para el desarrollo de nuevas aplicaciones y para una mayor optimización del trabajo con datos es la necesidad de habilidades y conocimientos específicos combinados, como pueden ser tecnológicos o matemáticos, entre otros.

3.2 Smart Objects

Un *Smart Object,* también conocido como *Intelligent Product*, es un elemento físico que posee diversas propiedades, identificable a lo largo de su vida útil, que va a interactuar con el entorno y con otros objetos. Además, mediante una conducta autónoma, puede actuar de manera inteligente según unas determinadas situaciones [1], [4], [5].

Los *Smart Objects* pueden ser clasificados en tres dimensiones en función del tipo de inteligencia [6], siendo posible determinar tanto el tipo de *Smart Object* como el de inteligencia, así como compararlo con otros.

Las dimensiones son las siguientes:

- Nivel de inteligencia: describe la inteligencia del objeto a través de tres categorías.
    - Gestión de la información: capacidad para manejar la información que recoge a través de sensores lectores u otras técnicas.
    - Notificación del problema: posibilidad de que sea capaz de notificar a su propietario cuando ha ocurrido un evento o problema determinado.
    - Toma de decisiones: capacidad propia de toma de decisiones sin intervención de un control externo.
- Localización de la inteligencia:
    - Inteligencia a través de la red: trata de que la inteligencia del objeto dependa totalmente de un agente externo al propio objeto como, por ejemplo, una red a la que se encuentre conectado.
    - Inteligencia en el objeto: albergan toda la inteligencia en ellos mismos.
- Agregación del nivel de inteligencia:
    - Inteligencia en el elemento: objetos que solo pueden manejar información, notificaciones y/o decisiones y, si tienen otros componentes, no pueden ser distinguidos como objetos individuales, como un sensor.
    - Inteligencia en el contendor: objetos que manejan información, notificaciones y/o decisiones. Es capaz de seguir funcionando como elemento u objeto a pesar de que se le desensamble alguna parte de él, como una placa *Arduino* con al menos tres sensores que, aunque se le quite un sensor, puede seguir funcionando como contenedor.

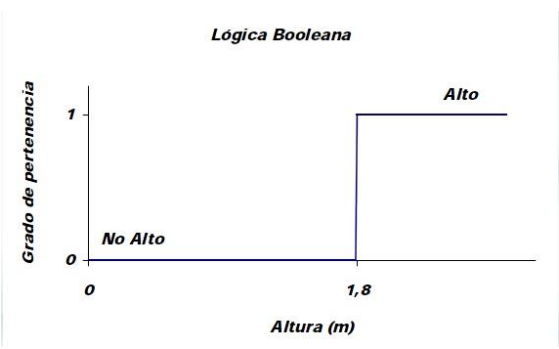
En este prototipo se usarán sensores con nivel de inteligencia de gestión de la información y de notificación del problema, con la localización de la inteligencia de gestión de la información y de notificación del problema, con la localización de la inteligencia a través de la red y de ambos niveles de agregación de nivel de inteligencia.

3.3 Lógica difusa.

Lofti A.Zadeh en 1964, fue quien introdujo por primera vez el concepto de conjunto borroso, se trataba de un profesor de la Universidad de California Berkeley que en un intento de representar y manipular datos que eran precisos. La *Teoría de los Subconjuntos Borrosos* fue mostrada posteriormente, ya que algunas magnitudes pueden tomar valores que difícilmente se pueden clasificar en un conjunto determinado. Esta teoría permite la definición adecuada de conjuntos que modelan las situaciones de imprecisión antes expuestas.

La lógica que utiliza expresiones que no son totalmente ciertas ni completamente falsas es la lógica difusa. Esta lógica se aplica a conceptos que puedan estar representados con un valor cualquiera de veracidad que se encuentre en un conjunto de valores que están entre dos extremos, la verdad absoluta y la falsedad total.

Por esto mismo se admite que se puedan presentar incertidumbres o imprecisiones, totalmente al contrario que ocurre con la lógica Booleana.



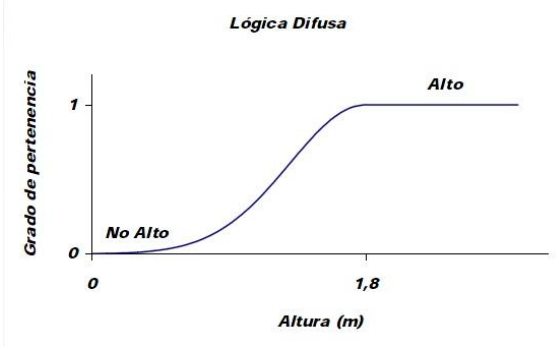


Imagen 3.3.1 Grafico Lógica Booleana *Imagen 3.3.2 Gráfico Lógica Difusa*

En lógica difusa se parte del hecho de que conceptos como alto, bajo, ruidoso, dulce, caro, amargo, barato, delgado, etc. son percibidos de manera diferente por cada persona. Por ejemplo, para una persona de que se encuentra en un hábitat de temperaturas muy frías como los Polos, el concepto de caliente puede ser arriba de 10 °C, mientras que para alguien que viva en una zona de temperaturas altas, caliente es arriba de 30 °C o en un proceso de fundición caliente es arriba de 300°C. Por esta razón los conjuntos CALIENTE, TIBIO y FRÍO son llamados *conjuntos difusos***.**

Un conjunto difuso es un conjunto con límites borrosos o "no muy bien" definidos.

Se considera un número borroso cuando es un caso concreto de un conjunto borroso que su función de pertenencia es continua, convexa y definida sobre un intervalo cerrado de los números reales, es decir un número que se encuentre en cualquier zona de la función que represente dicho conjunto borroso. Estas funciones pueden tener distintas formas siendo las más comunes las triangulares o trapezoidales que tienen un número reducido de parámetros, aunque también son habituales la gaussiana o la campana generalizada. A continuación se pueden verlas funciones de pertenencias comentadas.

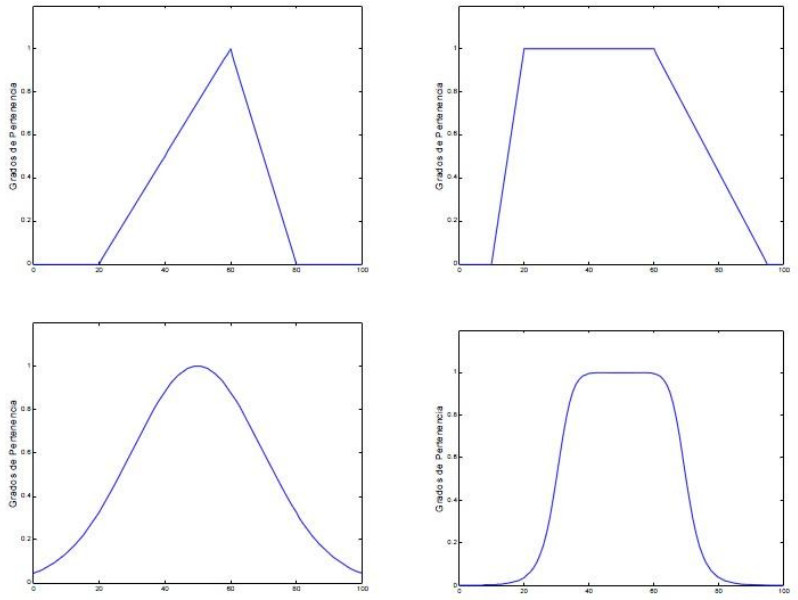


Imagen 3.3.2 Funciones de pertenencia.

Los conjuntos borrosos se pueden utilizar para representar variables lingüísticas cuyos valores son números borrosos que están definidos en términos lingüísticos. Al conjunto de estos números borrosos, que abarcan todo el universo de discurso de la variable, se le denomina *partición borrosa.*

El número de conjuntos borrosos que componen dicha partición se suele tomar según el grado de precisión requerido para esa variable. Tomar una gran cantidad de conjuntos borrosos (siete o más de siete, en general), tiene como ventaja el poder precisar las acciones que se van a llevar a cabo en el sistema en función de esta variable. La desventaja es que la *base de conocimiento* del sistema debe contemplar todos los términos lingüísticos (conjuntos), por lo que el tamaño de la misma crecerá enormemente.

Por ejemplo, si la temperatura se interpreta como una variable lingüística, el conjunto de valores que puede tomar podría ser {muy fría, fría, media, templada, cálida, calurosa}. Cada uno de estos términos está caracterizado por un conjunto borroso definido en el universo de discurso ([-6ºC, 48ºC]) de la variable temperatura, tal y como muestra la figura a continuación.

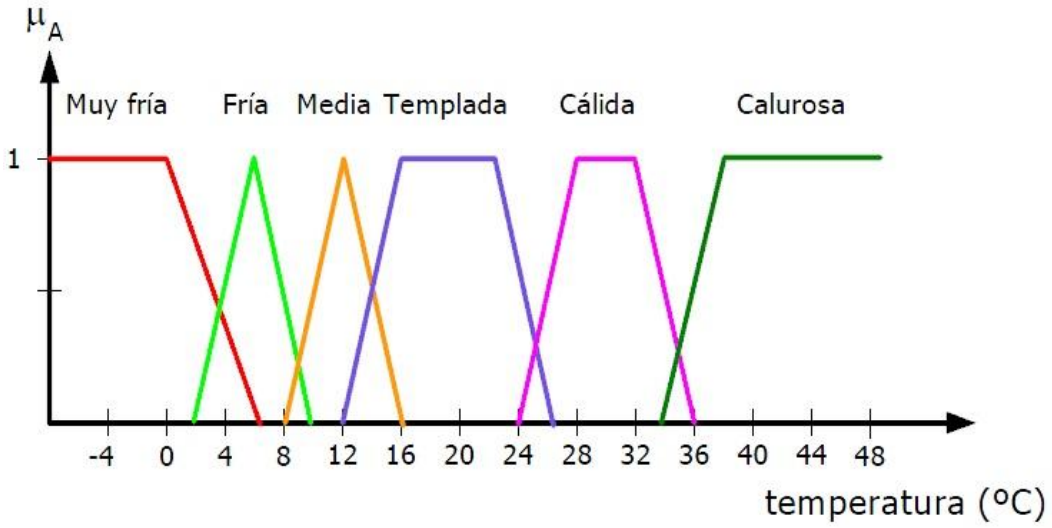


Imagen 3.3.3 Definición y partición borrosa de la variable lingüística temperatura.

A partir de la definición de las variables lingüística se definen las reglas difusas que son un conjunto de reglas SI-ENTONCES que puede ser definida de la siguiente forma:

- **SI** (x es A1) **Y** (y es B1) **ENTONCES** (z es C1)

El control difuso simula una acción humana intuitiva sobre la variable manipulada para llevar el proceso a la señal de consigna. Esta acción se realiza mediante las Reglas difusas, que utilizan las Variables lingüísticas, basadas en el conocimiento del comportamiento dinámico del proceso.

Las etapas de un controlador difuso:

- Entrada de datos físicos al controlador.

- Fuzzificación: Conversión de los datos físicos en variables difusas mediante las funciones de pertenencia. Se asignan los grados de pertenencia correspondiente a cada una de las etiquetas lingüísticas definidas.
- Inferencia difusa: Mediante la aplicación de reglas lingüísticas basadas en la experiencia a las variables difusas obtenidas anteriormente se obtienen las variables difusas manipuladas
- Defuzzificacion: Conversión de las variables difusas de salida en un valor concreto de la variable real de salida empleando métodos matemáticos.
- Salida de resultados físicos: Aplicación de la variable real de salida como acción de control.

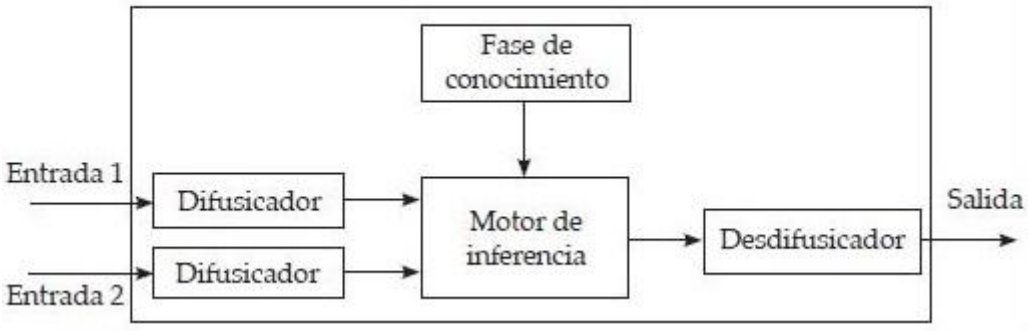


Imagen 3.3.4 Estructura de un controlador difuso.

3.4 Plataformas IoT

Actualmente, se pueden encontrar diversas plataformas para la interconexión de dispositivos, las cuales siguen creciendo a medida que pasa el tiempo.

En función de los servicios que ofrecen, se pueden distinguir varias clasificaciones. Algunas de ellas serán mencionadas en este artículo, mientras que otras serán explicadas más detalladamente puesto que ha sido posible probarlas.

Algunas de ellas han sido desarrolladas para el mundo empresarial, tal como puede ser *Xively*. Éstas son plataformas que permiten a las empresas aprovechar el *Internet de las cosas* para que se conecten de forma segura y robusta sus productos y usuarios, gestionen sus datos, se mejoren las satisfacciones de los clientes y generen nuevas fuentes de ingresos.

Otras plataformas, como *ThingSpeak* o *Paraimpu*, permiten crear canales a partir de un simple registro para que el usuario pueda, por una parte, ver y consumir los datos que se encuentran abiertos en varios formatos útiles como JSON, CSV o XML y, por otro lado, ofrecen la posibilidad de crear canales propios para compartir datos obtenidos de una forma particular. Estas posibilidades se ofrecen de forma gratuita registrándose en cada plataforma, aunque limitan el acceso a los usuarios mediante invitación o aprobación de la cuenta.

3.4.1 ThingSpeak

Ilustración 6 Logo ThingSpeak

ThingSpeak es una plataforma abierta para los productos y servicios de *Internet de las cosas* que permite a los usuarios interactuar con los objetos utilizando tecnologías Web estándar.

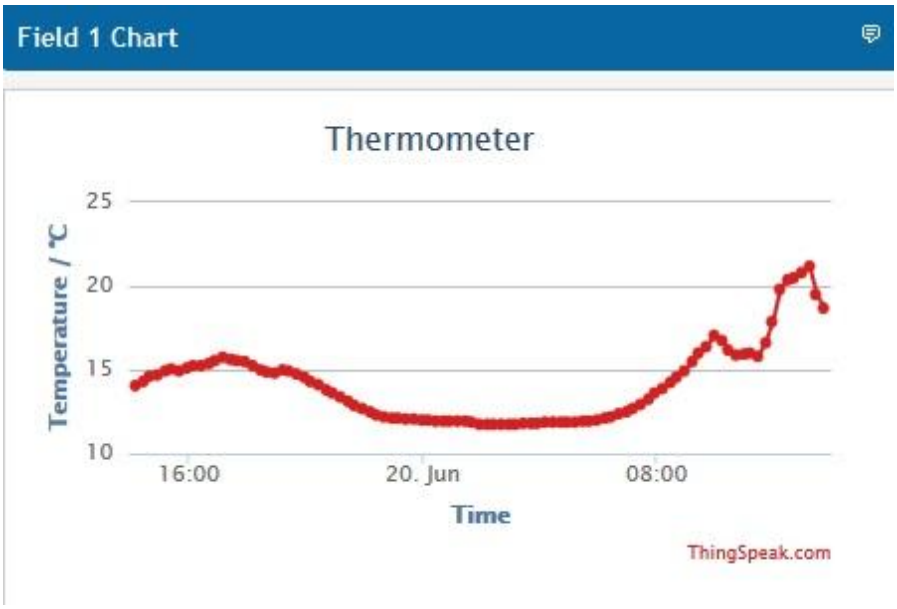


Ilustración 7 Gráfica de datos de ThingSpeak

Esta plataforma incluye el registro, procesado y distribución de la información, servicios basados en localización, integración con redes sociales, aplicaciones y plug-ins.

La creación de nuevos objetos y servicios se basa en la configuración del dispositivo seleccionado (*Arduino, Netduino, ioBridge*, etc.), colocando los datos de acceso de éste (ip, puerto, submáscara, etc.). De esta manera se permitirá crear un nuevo canal donde se van a mostrar los datos de los dispositivos conectados, gráficamente o mediante peticiones REST, para la obtención de los mismos en XML, CSV o JSON y así poder trabajar con ellos.

Ilustración 8 Cuadro de selección de muestra de datos


```
{
  ▾ channel: {
        id: 135,
        name: "Thermometer",
        description: "Wireless outdoor thermometer (Electric Imp, TI TMP102 sensor, 4 x AA Energizer L91).",
        latitude: "55.652072",
        longitude: "12.546301",
        field1: "Temperature",
        created_at: "2011-02-23T22:43:37Z",
        updated_at: "2015-06-20T11:59:55Z",
        elevation: "20m",
        last_entry_id: 49754
    },
  ▾ feeds: [
      ▾ {
            created_at: "2015-06-19T12:11:09Z",
            entry_id: 49659,
            field1: "14.0625"
        },
      ▾ {
            created_at: "2015-06-19T12:26:12Z",
            entry_id: 49660,
            field1: "14.3125"
        },
```


Ilustración 9 Ejemplo datos en formato JSON

3.4.2 Xively

Ilustración 10 Logo Xively

Plataforma de la compañía LOgMein conocida por aplicaciones de acceso remoto, *Xively* se define como una "Plataforma como un Servicio" (PassS) para *Internet de las cosas*. Para poder subir datos de los diferentes *Smart Objects* se debe de utilizar una *API Key* que proporcionan al registrarse o cuando se crea una nueva. De esta manera, es posible limitar algunas acciones REST a un dispositivo. Proporcionan una variedad de librerías en distintos lenguajes como *Java, Arduino o Android* para subir los datos. Cada usuario en su perfil puede añadir datos de distintos terminales, así como, a partir de crear un nuevo dispositivo, añadirle disparadores para que, después de cumplirse una determinada condición, realice peticiones Post a un servicio HTTP mediante un método REST [7].

3.4.3 Paraimpu

Ilustración 11 Logo Paraimpu

Plataforma que permite conectar a la Web *Smart Objects*, servicios y dispositivos y proporciona API´s para crear aplicaciones personalizadas. Permite que los sensores, motores, actuadores, micro controladores como *Arduino*, electrodomésticos, sistemas de iluminación y domótica puedan interactuar entre sí y conectarse a la Web.

Facilita la composición e interconexión de sus objetos y permite vincularlos a las redes sociales como *Facebook* y *Twitter*. Ofrece la posibilidad de manejar de forma remota su hogar, sus aplicaciones, así como configurar sus disparadores configurados para que reaccionen a eventos establecidos con reglas.

Paraimpu es una herramienta social que, además de comunicarse con sus redes, permite compartir cosas con los amigos dejando que otras personas usen los datos producidos y los objetos para crear sus propias conexiones y dispositivos.

3.4.4 Midgar

Midgar es una plataforma de Internet of Things que genera aplicaciones interconectores de objetos heterogéneos y ubicuos.

El usuario con el Lenguaje de Dominio Específico (DSL) que presenta, podrá interconectar diferentes objetos de una manera sencilla usando un editor gráfico.

A través de este DSL el usuario podrá describir el flujo que desee que realice la aplicación, de manera que podrá conectar multitud de objetos y leer de ellos cualquier parámetro de un servicio que ellos posean y del que suban datos. Así el usuario podrá leer datos y dependiendo de los datos, mandar realizar ciertas acciones a los mismos objetos o a otros diferentes.

Estas acciones difieren entre unos objetos u otros, se registran al mismo tiempo que los objetos y los servicios de manera que un objeto puede tener tantas acciones como servicios se quiera, igual que una aplicación puede interconectar distintos objetos conectados y acciones a realizar.[3]

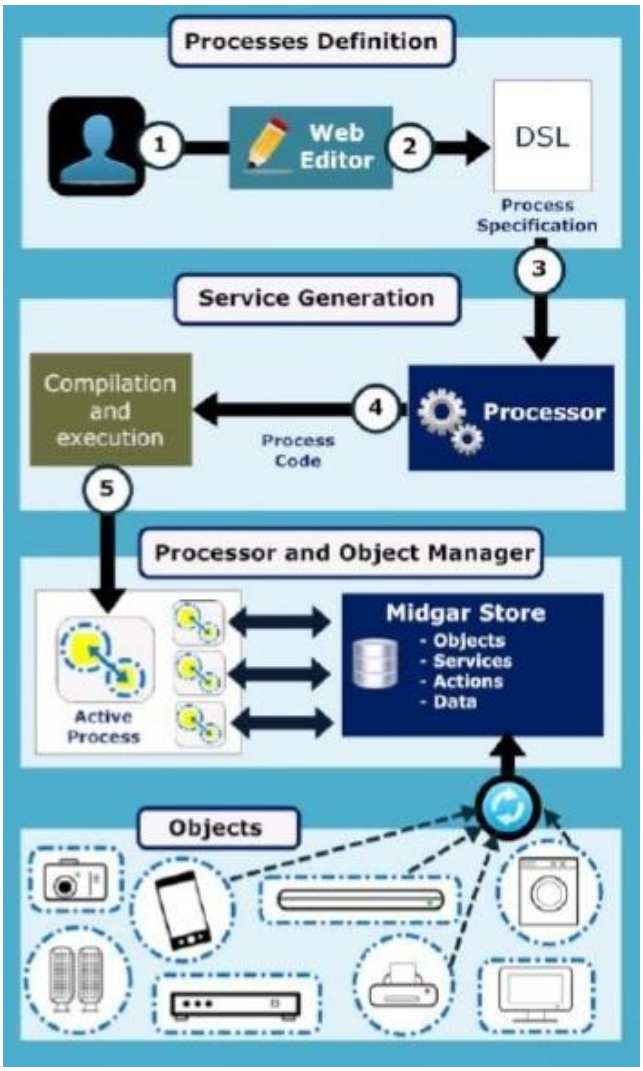


Ilustración 12 Arquitectura plataforma MIDGAR

3.5 Hardware

3.5.1 Arduino

Arduino es una plataforma de hardware libre, que consta de una placa con un micro controlador y un entorno de desarrollo creado para facilitar el uso de electrónica.

Arduino puede ser utilizado para desarrollar objetos interactivos, tomando como entradas desde una gran variedad de sensores o *switches* y controlando una gran variedad de luces, motores u otras salidas físicas. Los proyectos de *Arduino* pueden ser solitarios sin conectarse a ningún PC, o pueden estar comunicados con un software ejecutado en un ordenador. Las placas pueden ensamblarse a mano o ser obtenidas pre ensambladas; el IDE de código abierto se puede descargar libremente.

Su lenguaje de programación es una implementación Wiring, es decir, una plataforma computacional física similar.

Arduino es una placa relativamente barata comparada con otros microcontroladores, con un software que está preparado para trabajar en Windows, Macintosh OSX, y Linux.

Como especificaciones técnicas, es destacable mencionar que *Arduino Uno* es una placa basada en ATmega328 (datasheet). Cuenta con: 14 pines digitales de entrada y salida (6 de los cuales pueden ser utilizados como salidas PWM), 6 entradas analógicas, un resonador cerámico de 16 MHz, conexión USB, conector de alimentación, una cabecera ICSP y un botón de reinicio.

A continuación se muestran algunas de las distintas placas que Arduino posee con algunas especificaciones técnicas.

3.5.1.1 *Arduino UNO*

ARDUINO UNO	
Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz
Length	68.6 mm
Width	53.4 mm
Weight	25 g

Tabla 1 Especificaciones Técnicas Arduino Uno

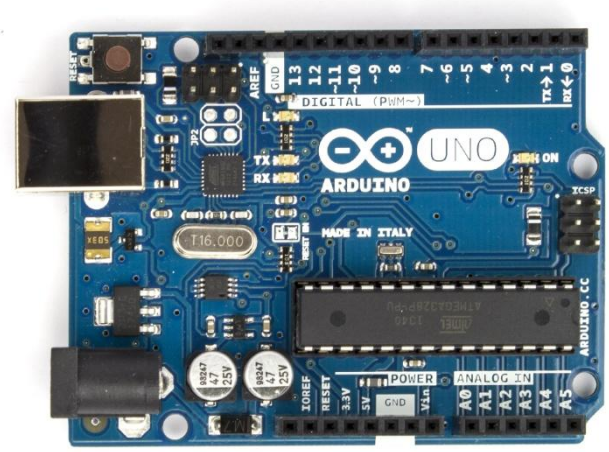


Ilustración 13 Arduino UNO

3.5.1.2 *Arduino Mega*

ARDUINO MEGA 2560	
Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	54 (of which 15 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz

Tabla 2 Especificaciones técnicas Arduino Mega 2560

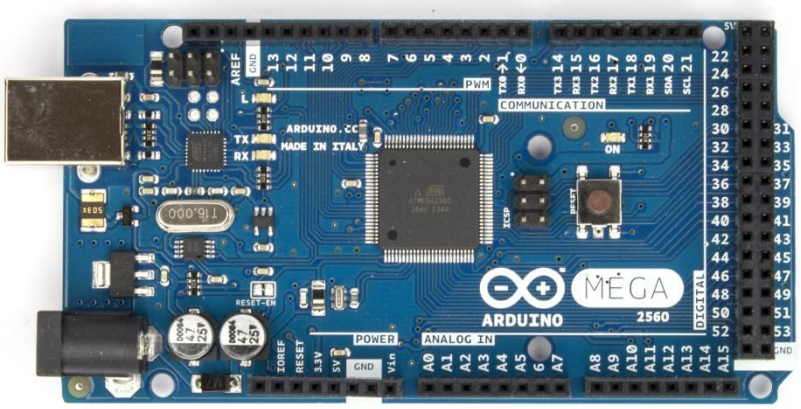


Ilustración 14 Arduino Mega 2560

3.5.1.3 *Arduino Nano*

ARDUINO UNO	
Microcontroller	Atmel ATmega168 or ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	8
DC Current per I/O Pin	40 mA
Flash Memory	16 KB (ATmega168) or 32 KB (ATmega328) of which 2

	KB used by bootloader
SRAM	1 KB (ATmega168) or 2 KB (ATmega328)
EEPROM	512 bytes (ATmega168) or 1 KB (ATmega328)
Clock Speed	16 MHz
Length	45 mm
Width	18 mm
Weight	5 g

Tabla 3 Especificaciones Arduino Nano

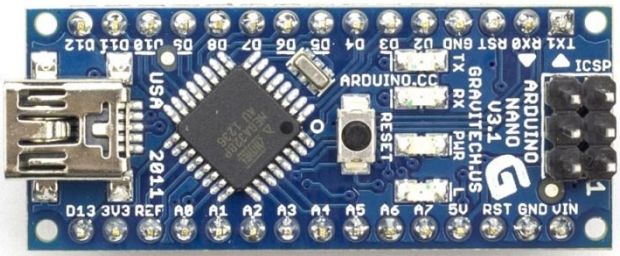


Ilustración 15 Arduino Nano

3.6 Metodología

3.6.1 Métrica 3

La métrica 3 es una metodología de planificación, desarrollo y mantenimiento de sistemas de información promovida por el Ministerio de Hacienda y Administraciones Públicas.

Se puede utilizar libremente siempre y cuando se cite la fuente de su propiedad intelectual: el Ministerio de Administraciones Públicas.

Esta metodología ofrece la sistematización de actividades para dar soporte al ciclo de vida del software.

Objetivos que permite alcanzar esta métrica son:

- Proporcionar o definir Sistemas de Información
- Dotar a la Organización de productos software que satisfagan las necesidades de los usuarios dando una mayor importancia al análisis de requisitos.
- Mejorar la productividad de los departamentos de Sistemas y Tecnologías de la Información y las Comunicaciones, permitiendo una mayor capacidad de adaptación a los cambios y teniendo en cuenta la reutilización en la medida de lo posible.

- Facilitar la comunicación y entendimiento entre los distintos participantes en la producción de software a lo largo del ciclo de vida del proyecto, teniendo en cuenta su papel y responsabilidad, así como las necesidades de todos y cada uno de ellos.
- Facilitar la operación, mantenimiento y uso de los productos software obtenido.

3.6.2 Extreme programming- XP

Extreme Programming es uno los procesos ágiles de desarrollo más populares que hay, mostrando su éxito en distintos proyectos de empresas en todo el mundo.

Esta programación tiene éxito porque hace hincapié en la satisfacción del cliente. En lugar de entregar todo lo que se pueda en una fecha marcada en un futuro lejano este proceso proporciona el software que se necesita exactamente cuándo se necesita, ya que sus desarrolladores responden con eficiencia a las cambiantes necesidades del cliente durante el ciclo de vida del desarrollo.

Este proceso enfatiza el trabajo en equipo, los gerentes clientes y desarrolladores están en contacto para configurar un equipo de colaboración. [14]

Los programadores de XP:

- Se comunican constantemente con sus clientes y compañeros
- Mantienen un diseño simple y limpio
- Reciben retroalimentación probando su Software a partir del primer día
- Entregan el sistema al producto a los clientes tan pronto como sea posible para implementar los cambios que sugieran.

En este caso se ha llevado una programación que quizás no hay sido exactamente XP, pero sí que ha estado basada en esta metodología.

Una de las causas por las que se dice que está basado en esta metodología es porque el programador en este caso solo ha sido una persona, aunque por otro lado las similitudes son varias, como el continuo proceso de pruebas, reuniones semanales con el co-director del proyecto que representaba el cliente y el diseñar para cada momento lo que se iba pidiendo.

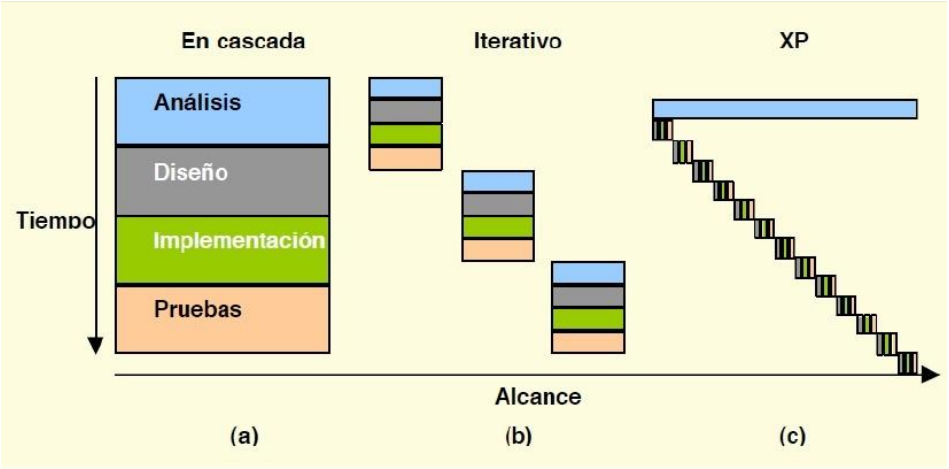


Ilustración 16 Ciclos de desarrollo de tres metodologías diferentes

Capítulo 4. Planificación del Proyecto y Resumen de Presupuestos

4.1 Planificación

La primera imagen muestra la planificación inicial así como su diagrama de Gantt asociado. Esta planificación fue desarrollada al inicio del desarrollo del proyecto, calculando aproximadamente la duración de cada tarea desglosada.

A lo largo del desarrollo del trabajo, como era previsible, se fue modificando dicha planificación a medida que surgían problemas o a medida que los directores del máster indicaron cambios en la planificación y rediseños.

	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras	Nombres de los recursos
1	Trabajo Fin De Máster	141 días	mar 20/01/15	mar 04/08/15		
2	Evaluación de Información	4 días	mar 20/01/15	vie 23/01/15		
8	Análisis	11 días	lun 26/01/15	lun 09/02/15	2	
9	Plataforma	4 días	lun 26/01/15	jue 29/01/15		
10	Análisis Software Existente	2 días	lun 26/01/15	mar 27/01/15		Director Proyecto Empresa
11	Análisis de Requisitos	2 días	mié 28/01/15	jue 29/01/15	10	Director Proyecto Empresa
12	Fi-Ware	7 días	vie 30/01/15	lun 09/02/15	9	
13	General Enablers	4 días	vie 30/01/15	mié 04/02/15		Ricardo Diez;Director Proy
14	Specific Enablers	2 días	vie 30/01/15	lun 02/02/15		Ricardo Diez;Director Proy
15	Creación Enablers	3 días	jue 05/02/15	lun 09/02/15	13	Ricardo Diez;Director Proy
16	DNN	3 días	jue 05/02/15	lun 09/02/15	9	
19	Cloud Computing	2 días	jue 05/02/15	vie 06/02/15	13	
22	Diseño	18 días	mar 10/02/15	jue 05/03/15	8	
23	Fi-Ware	12 días	mar 10/02/15	mié 25/02/15		
24	Implemetación General I	6 días	mar 10/02/15	mar 17/02/15		
28	Specific Enablers Propio	6 días	mié 18/02/15	mié 25/02/15	24	
31	DNN	6 días	jue 26/02/15	jue 05/03/15	23	
37	Implementación	79 días	vie 06/03/15	mié 24/06/15	22	
38	Fi-Ware	64 días	vie 06/03/15	mié 03/06/15		
53	DNN	15 días	jue 04/06/15	mié 24/06/15	38	
59	Pruebas	6 días	jue 25/06/15	jue 02/07/15	37	
63	Documentación	23 días	vie 03/07/15	mar 04/08/15	59	
64	Documentación de Entrega	23 días	vie 03/07/15	mar 04/08/15		

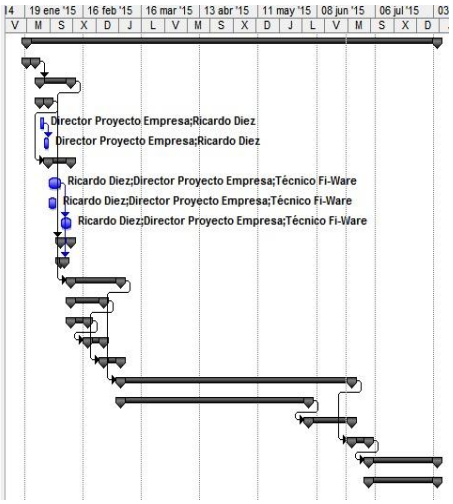


Ilustración 17 Planificación y diagrama de Gantt Inicial

A continuación muestra la planificación final con respecto al desarrollo del proyecto, aquí se puede observar que existen diferencias con respecto a la planificación inicial producidos por los factores que se mencionaron en el párrafo anterior.

- **Evaluación de Información:** Presentación del proyecto propuesto, así como primeras impresiones sobre las tecnologías propuestas a desarrollar y elección del mismo.
- **Análisis**: Estudio de la documentación aportada por el director del proyecto y por el co-director y de las distintas fuentes de información que se pueden encontrar.

Trabajo Fin De Máster	**111 días**	**lun 02/02/15**	**lun 06/07/15**		
Evaluación de Información	**15 días**	**lun 02/02/15**	**vie 20/02/15**		
Presentación del Proyecto	6 días	lun 02/02/15	lun 09/02/15		Ricardo Diez;Co-Director Proyect
Informe de la tecnología Fi-Ware	9 días	mar 10/02/15	vie 20/02/15	3	Co-Director Proyecto ;Ricardo Die
Informe Azure Cloud Computing	4 días	mar 10/02/15	vie 13/02/15	3	Co-Director Proyecto ;Ricardo Die
Informe CMS DNN	3 días	mar 10/02/15	jue 12/02/15	3	Co-Director Proyecto ;Ricardo Die
Elección del Proyecto	3 días	mar 10/02/15	jue 12/02/15	3	Ricardo Diez;Co-Director Proyect
Análisis	**23 días**	**lun 23/02/15**	**mié 25/03/15**	**2**	
Plataforma	**5 días**	**lun 23/02/15**	**vie 27/02/15**		
Análisis Software Existente	4 días	lun 23/02/15	jue 26/02/15		Co-Director Proyecto ;Ricardo Die
Análisis de Requisitos	5 días	lun 23/02/15	vie 27/02/15		Co-Director Proyecto ;Ricardo Die
Fi-Ware	**18 días**	**lun 02/03/15**	**mié 25/03/15**	**9**	
General Enablers	6 días	lun 02/03/15	lun 09/03/15		Ricardo Diez;Co-Director Proyect
Specific Enablers	6 días	mar 10/03/15	mar 17/03/15	13	Ricardo Diez;Co-Director Proyect
Creación Enablers	6 días	mié 18/03/15	mié 25/03/15	14	Ricardo Diez;Co-Director Proyect

Ilustración 18 Diagrama de tareas del proyecto-1

- **Diseño:** Diseño del proyecto desglosado en las tecnologías a usar y diferenciado entre las partes que se han de crear de cero y las que se pueden modificar.

- E**valuación de Problemas encontrados:** Se evaluaron los problemas encontrados en el desarrollo del proyecto marcado en el inicio.

Diseño	**26 días**	**jue 26/03/15**	**jue 30/04/15**	**8**	
Fi-Ware	**26 días**	**jue 26/03/15**	**jue 30/04/15**		
Implemetación General Enablers	**26 días**	**jue 26/03/15**	**jue 30/04/15**		
KeyRock	13 días	jue 26/03/15	lun 13/04/15		Co-Director Proyecto ;Ricardo Die
Orion Context Broker	13 días	mar 14/04/15	jue 30/04/15	19	Ricardo Diez;Co-Director Proyect
Cosmos	13 días	jue 26/03/15	lun 13/04/15		Co-Director Proyecto ;Ricardo Die
Evaluación de Problemas encontrados	**8 días**	**vie 01/05/15**	**mar 12/05/15**	**16**	
Problemas tecnología Fi-Ware	8 días	vie 01/05/15	mar 12/05/15		Director Proyecto Universidad;Ric
Problemas integración General Enablers	7 días	vie 01/05/15	lun 11/05/15		Director Proyecto Universidad;Ric
Valoración cambio de tecnología	5 días	vie 01/05/15	jue 07/05/15		Director Proyecto Universidad;Ric

Ilustración 19 Diagrama de tareas del proyecto-2

- **Rediseño del TFM**: Después de hacer una valoración de los problemas encontrados se realizó un Rediseño del trabajo fin de máster que esta vez estuviera enfocado en algo con mayor abanico de soluciones.
- **Análisis:** Análisis de los informes sobre los aspectos que se va a trabajar en el trabajo fin de máster.
- **Diseño:** Diseño del proyecto donde se escogieron las distintas tecnologías a usar.

Rediseño del TFM	**39 días**	**mié 13/05/15**	**lun 06/07/15**	**22**	
Evaluación del Prototipo	**4 días**	**mié 13/05/15**	**lun 18/05/15**		
Estudio del Internet de las cosas	4 días	mié 13/05/15	lun 18/05/15		Ricardo Diez
Alcance del prototipo a realizar	2 días	mié 13/05/15	jue 14/05/15		Ricardo Diez;Co-Director Proyect
Análisis	**3 días**	**mar 19/05/15**	**jue 21/05/15**	**27**	
Informe IoT	**3 días**	**mar 19/05/15**	**jue 21/05/15**		**Ricardo Diez;Co-Director Proy**
Informe Smart Objects	2 días	mar 19/05/15	mié 20/05/15		
Informe micro-ordenadores	2 días	mar 19/05/15	mié 20/05/15		
Informe micro-controladores	2 días	mar 19/05/15	mié 20/05/15		
Informe Sensores	3 días	mar 19/05/15	jue 21/05/15		
Inorme Lógica Difusa	2 días	mar 19/05/15	mié 20/05/15		
Especificación de Requisitos	3 días	mar 19/05/15	jue 21/05/15		
Diseño	**3 días**	**vie 22/05/15**	**mar 26/05/15**	**31**	**Ricardo Diez;Co-Director Proy**
Elección de micro-ordenador	2 días	vie 22/05/15	lun 25/05/15		Raspberry[1]
Elección de micro-controlador	3 días	vie 22/05/15	mar 26/05/15		Arduino[1]
Elección redes IoT	2 días	vie 22/05/15	lun 25/05/15		
Diseño Prototipo	2 días	vie 22/05/15	lun 25/05/15		

Ilustración 20 Diagrama de tareas del proyecto-3

- **Módulos**: Desglose de los distintos módulos en los que se dividió el proyecto a la hora de trabajar en el prototipo.
    - **Raspberry**
    - **Arduino**
    - **Redes IoT**

Módulos	**25 días**	**mié 27/05/15**	**mar 30/06/15**	**38**	**Ricardo Diez;Co-Director Proy**
Raspberry	**5 días**	**mié 27/05/15**	**mar 02/06/15**		
Instalacion	**2 días**	**mié 27/05/15**	**jue 28/05/15**		
Sistema Operativo	2 días	mié 27/05/15	jue 28/05/15		
IDE Java	2 días	mié 27/05/15	jue 28/05/15		
Controladores	2 días	mié 27/05/15	jue 28/05/15		
Sensores	2 días	mié 27/05/15	jue 28/05/15		
Implementación	**3 días**	**vie 29/05/15**	**mar 02/06/15**	**45**	
Lectura de sensores	2 días	vie 29/05/15	lun 01/06/15		
Control de actuadores	3 días	vie 29/05/15	mar 02/06/15		
Arduino	**4 días**	**mié 03/06/15**	**lun 08/06/15**	**44**	
Instalación	**2 días**	**mié 03/06/15**	**jue 04/06/15**		
Sensores	2 días	mié 03/06/15	jue 04/06/15		
Implementación	**2 días**	**vie 05/06/15**	**lun 08/06/15**	**54**	
Lectura de sensores	2 días	vie 05/06/15	lun 08/06/15		
Subida de datos a Midgar	2 días	vie 05/06/15	lun 08/06/15		
Redes Iot	**2 días**	**mar 09/06/15**	**mié 10/06/15**	**53**	
Lectura de datos	2 días	mar 09/06/15	mié 10/06/15		
Parseo de datos	2 días	mar 09/06/15	mié 10/06/15		

Ilustración 21 Diagrama de tareas del proyecto-4

    - **Lógica Difusa**

Lógica difusa	**14 días**	**jue 11/06/15**	**mar 30/06/15**	**59**	
Sensación térmica	**6 días**	**jue 11/06/15**	**jue 18/06/15**		
Diseño	**2 días**	**jue 11/06/15**	**vie 12/06/15**		
Variables Lingüísticas	2 días	jue 11/06/15	vie 12/06/15		
Reglas Difusas	2 días	jue 11/06/15	vie 12/06/15		
Implementación	2 días	lun 15/06/15	mar 16/06/15	64	
Cálculo sensación térmica	2 días	mié 17/06/15	jue 18/06/15	67	
Salida Final	**8 días**	**vie 19/06/15**	**mar 30/06/15**	**68**	
Diseño	**4 días**	**vie 19/06/15**	**mié 24/06/15**		
Variables lingüísticas	2 días	vie 19/06/15	lun 22/06/15	63	
Reglas difusas	2 días	mar 23/06/15	mié 24/06/15	71	
Implementación	2 días	jue 25/06/15	vie 26/06/15	72	
Cálculo Salida Final	2 días	lun 29/06/15	mar 30/06/15	73	

Ilustración 22 Diagrama de tareas del proyecto-5

- **Pruebas**: Realización de pruebas para comprobar el funcionamiento deseado de los elementos con distintos tipos de obtención de datos.
- **Documentación**: Documentación del prototipo.

Pruebas	**2 días**	**mié 01/07/15**	**jue 02/07/15**	**69**	**Director Proyecto Universida**
Pruebas Unitarias	2 días	mié 01/07/15	jue 02/07/15		
Datos rales	2 días	mié 01/07/15	jue 02/07/15		
Datos semi-aleatorios	2 días	mié 01/07/15	jue 02/07/15		
Documentación	**4 días?**	**mié 01/07/15**	**lun 06/07/15**	**43**	**Ricardo Diez;Co-Director Proy**
Desarrollo de la documentación	2 días	mié 01/07/15	jue 02/07/15		
Revisión	1 día	vie 03/07/15	vie 03/07/15	80	
Entrega de la documentación	1 día	lun 06/07/15	lun 06/07/15	81	
	1 día?	mié 01/07/15	mié 01/07/15		

Ilustración 23 Diagrama de tareas del proyecto-6

4.2 Resumen del Presupuesto

A continuación se muestra una tabla que muestra el resumen del presupuesto.

Se estima unos costes de desarrollo de 37748,68 €.

Categoría	Importe
Recursos Humanos	35978.04
Recursos materiales	728.80 €
Subtotal	36.706,84 €
IVA (21%)	7708,43 €
TOTAL	44.415,27€

Tabla 4 Resumen del presupuesto

Capítulo 5. Análisis

5.1 Definición del Sistema

5.1.1 Determinación del Alcance del Sistema

Este prototipo tiene como fin simular el control de temperatura en un lugar determinado de manera automatizada. La salida del control de este prototipo se calculará mediante lógica difusa para darle una inteligencia que permita gestionarse sin necesidad de un control humano constante.

Se tomarán tres valores distintos con el fin de obtener un mayor control de la temperatura y conseguir unas condiciones ambientales interiores propuestas. Para la captura de estos valores se usará la tecnología de Internet de las Cosas accediendo a los datos de cada valor de una manera diferente para abarcar algunas de las distintas vías con las que podemos acceder a datos recogidos por sensores.

El prototipo calculará mediante lógica difusa la sensación térmica en el exterior con respecto a la humedad y temperatura obtenida. Esta sensación térmica se obtendrá a través de unas reglas difusas previamente definidas que toman como base la tabla de valores de sensaciones térmicas que a continuación se muestran y que tienen como entradas, la humedad y la temperatura.

TEMPERATURA (° C)	HUMEDAD RELATIVA (%)																			
	0	5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80	85	90	95	100
20	16	16	17	17	17	18	18	19	19	19	19	19	20	20	20	21	21	21	21	21	21
21	18	18	18	19	19	19	19	19	20	20	20	20	21	21	21	22	22	22	22	22	23
22	19	19	19	20	20	20	20	20	21	21	21	21	22	22	22	22	23	23	23	23	24
23	20	20	20	20	21	21	22	22	22	23	23	23	23	24	24	24	24	24	24	25	25
24	21	21	22	22	22	22	23	23	23	24	24	24	24	25	25	25	25	26	26	26	26
25	22	23	23	23	24	24	24	24	24	24	25	25	25	26	26	26	27	27	27	28	28
26	24	24	24	24	25	25	25	26	26	26	26	27	27	27	27	28	28	29	29	29	30
27	25	25	25	25	26	26	26	27	27	27	27	28	28	29	29	30	30	31	31	31	33
28	26	26	26	26	27	27	27	28	28	28	29	29	29	30	31	32	32	33	34	34	36
29	26	26	27	27	27	28	29	29	29	29	30	30	31	33	33	34	35	35	37	38	40
30	27	27	28	28	28	28	29	29	30	30	31	32	33	34	35	36	37	39	40	41	45
31	28	28	29	29	29	29	30	31	31	31	33	34	35	36	37	39	40	41	45	45	50
32	29	29	29	29	30	31	31	33	33	34	35	35	37	39	40	42	44	45	51	51	55
33	29	29	30	30	31	33	33	34	34	35	36	38	39	42	43	45	49	49	53	54	55
34	30	30	31	31	32	34	34	35	36	37	38	41	42	44	47	48	50	52	55		
35	31	32	32	32	33	35	35	37	37	40	40	44	45	47	51	52	55				
36	32	33	33	34	35	36	37	39	39	42	43	46	49	50	54	55					
37	32	33	34	35	36	38	38	41	41	44	46	49	51	55							
38	33	34	35	36	37	39	40	43	44	47	49	51	55								
39	34	35	36	37	38	41	41	44	46	50	50	55									
40	35	36	37	39	40	43	43	47	49	53	55										
41	35	36	38	40	41	44	45	49	50	55											
42	36	37	39	41	42	45	47	50	52	55											
43	37	38	40	42	44	47	49	53	55												
44	38	39	41	44	45	49	52	55													
45	38	40	42	45	47	50	54	55													
46	39	41	43	45	49	51	55														
47	40	42	44	47	51	54	55														
48	41	43	45	49	53	55															
49	42	45	47	50	54	55															
50	42	45	48	50	55																

Ilustración 24 Tabla de sensaciones térmicas

Los valores exteriores se obtienen de dos maneras distintas:

- En el caso de la Humedad exterior, se accederá a una de las plataformas de Internet de las cosas que existe, en este caso será *ThingSpeak* que ofrece datos de distintas regiones en varios formatos. JSON es el tipo escogido en esta ocasión, por lo que una vez parseado correctamente tendremos la primera entrada para el cálculo con lógica difusa de la sensación térmica.
- Con la temperatura Exterior, accederemos a un microcontolador Arduino que tiene conectado un sensor de temperatura cuyos datos se suben a otra plataforma de internet de las cosas que en este caso es MIDGAR. MIDGAR se encarga de recoger los datos del Arduino, para ofrecerlos en JSON y así tener la segunda entrada para el cálculo de la sensación térmica.

Una vez calculada la sensación térmica, el siguiente paso será obtener la temperatura local de la sala que se quiere calcular.

- La temperatura interior, se obtendrá a través de un sensor conectado al microordenador Raspberry que será desde donde estaremos ejecutando el prototipo.

Así para la obtención de la salida de los controladores de temperatura, ejecutaremos otro bloque de lógica difusa con las dos temperaturas obtenidas, la interior y la sensación térmica.

La salida producida en este bloque de lógica difusa será la que determine el funcionamiento de los elementos de control de la temperatura.

La salida estará definida en cinco estados posibles, cada uno de ellos simulará el control de temperatura con un conjunto de Led´s's determinados de la siguiente manera.

- Dos Led's's rojos para simular el funcionamiento de una calefacción. Los dos Led's rojos luciendo representan la máxima potencia y un solo LED rojo encendido representa potencia normal, siempre asignado por la salida final.
- Dos Led's amarillos para simular el funcionamiento de una aire acondicionado. Los dos Led's amarillos luciendo representan la máxima potencia y un solo LED amarillo encendido representa potencia normal.
- Un Led's tricolor que representa que el prototipo tiene una temperatura adecuada y ningún aparato estará en funcionamiento.

5.2 Requisitos del Sistema

5.2.1 Obtención de los Requisitos del Sistema

5.2.1.1 *Explicación textual*

Se va a desarrollar un prototipo que simulará el control de temperatura de un determinado lugar de una manera automatizada, por lo que se le asignará a dicho prototipo de una lógica difusa para que lleve a cabo su función.

Este prototipo está basado en Internet de las Cosas, de manera que se consumirán datos de distintos objetos inteligentes y plataformas de internet de las cosas.

Algunos de estas plataformas ya están creadas y solo debemos de consumir los datos que tienen abiertos a cualquier usuario aunque por otro lado, se subirán datos deseados a alguna de estas plataformas existentes para su consumo posterior.

Se instalarán distintos sensores en microcontroladores y microordenadores para obtener datos que son fundamentales en el prototipo, así también se controlarán otros actuadores instalados en un microordenador que servirá para representar las salidas finales.

Para el funcionamiento del prototipo el usuario, que deberá de tener un cierto conocimiento sobre internet de las cosas y sensores, arrancará el prototipo encontrando varias opciones de uso dependiendo de si quiere una simulación con los datos reales que se están generando en tiempo real, o si por otro lado quiere ver alguna simulación ficticia con datos generados cuasi aleatoriamente.

5.2.1.2 *Requisitos funcionales*

Código	Nombre Requisito	Descripción del Requisito
R1	Usuario anónimo	
R1.1	Ejecutar Hilo	Se ejecutará el programa constantemente y obtendremos una simulación con datos en tiempo real.
R1.2	Ejecutar valores unitarios	Se ejecutará el programa pidiendo datos concretos al usuario por pantalla.
R1.3	Ejecutar Array de tamaño introducido por usuario	Se ejecutará el programa con datos aleatorios creados en un Array con longitud introducida por el usuario.
R1.4	Ejecutar Array Semialeatorios	Se ejecutará el programa con datos semialeatorios, que simularán las subidas y bajadas de temperatura y humedad.
R1.5	Ejecutar hilo con Valores aleatorios	Se ejecutará el programa ininterrumpidamente con datos semialeatorios que tendrán valores similares.
R2	Subir Datos a Midgar	Se obtendrán los datos del Arduino y se subirán a Midgar.
R2.1	Obtener datos desde Midgar	Consumir Datos desde la red Midgar.
R2.2	Obtener datos desde Raspberry	Obtener datos que ofrezca el sensor de temperatura instalado en la Raspberry.
R2.3	Obtener datos desde ThingSpeak	Obtener datos desde la red IoT ThingSpeak.
R3	Calculo de sensaciones térmicas exteriores	Se calcularán las sensaciones térmicas con respecto a la humedad y temperatura obtenida a través de lógica difusa.
R3.1	Cálculo de la señal final	Se calculará la señal final del prototipo con respecto a la sensación térmica exterior y la temperatura interior a través de lógica difusa.
R4	Representación de la salida final	Representación de la salida final de la simulación del control de temperatura.
R4.1	Manejo de actuadores	Control y manejos de los actuadores instalados.

Tabla 5 Requisitos funcionales

5.2.1.3 *Requisitos no funcionales*

Código	Nombre Requisito	Descripción del Requisito
R5	Requisitos de usuario	
R5.1	Conocimientos sobre Internet de las cosas	Los usuarios deberán tener un conocimiento sobre sensores, objetos inteligentes o microcontroladores.
R6	Requisitos tecnológicos	
R6.1	Uso de Raspberry	Se trabajará desde una Raspberry Pi 2, que tiene instalado Raspbian como sistema operativo.
R6.2	Uso de Arduino	Su usará un Arduino Uno con su IDE de desarrollo para la obtención de datos.
R7	Requisitos de tiempo de respuesta	
R7.1	Accesibilidad 24/7	El sistema deberá estar en funcionamiento, las 24

		horas los 7 días de las semana

Tabla 6 Requisitos no funcionales

5.2.2 Identificación de Actores del Sistema

En este prototipo se puede identificar solamente un tipo de actor, se trata de un usuario no identificado, es decir cualquier persona que esté en contacto con el prototipo y pueda ejecutarlo con los conocimientos adecuados.

Este usuario una vez ejecutado el prototipo pondrá algunos servicios secundarios en marcha para el funcionamiento del mismo, los cuales van a ser especificados a continuación.

5.2.3 Especificación de Casos de Uso

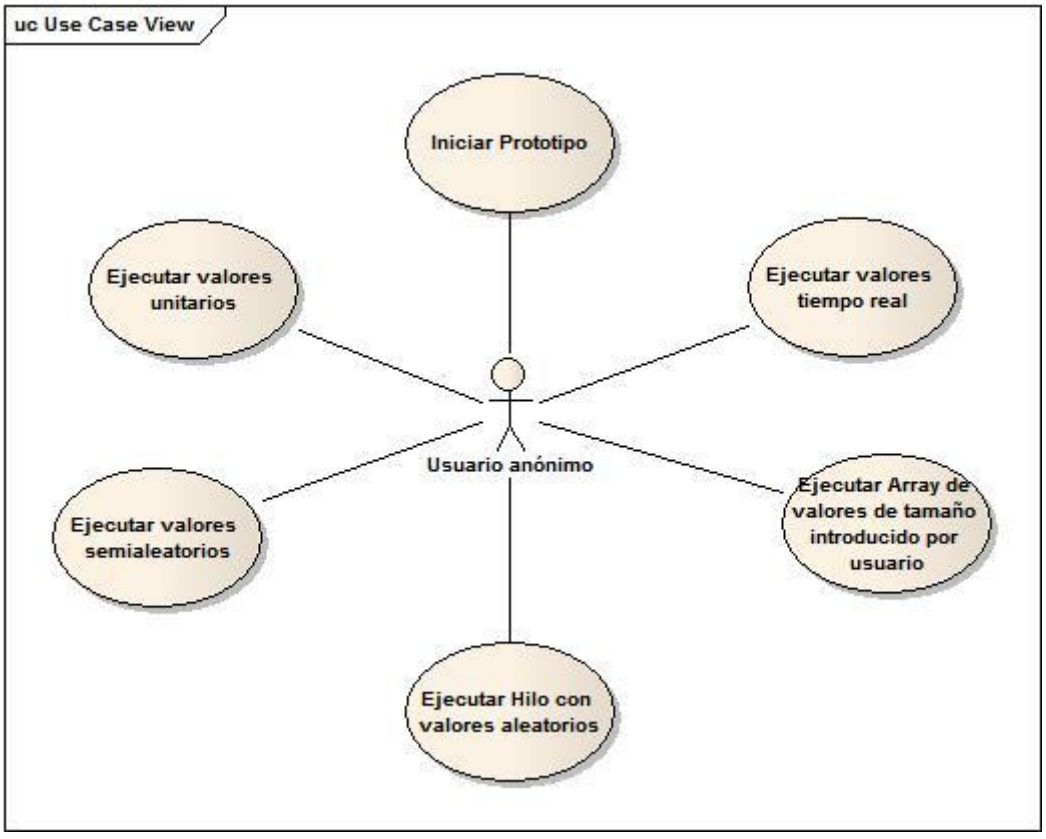


Ilustración 25 Casos de Uso

Nombre del Caso de Uso
Iniciar prototipo
Descripción
El usuario pondrá en marcha el prototipo para escoger el tipo de ejecución que desea realizar posteriormente.

Tabla 7 Caso de uso Iniciar prototipo

Nombre del Caso de Uso
Ejecutar valores tiempo real
Descripción
El usuario ejecutará el programa de modo que se obtendrán valores en tiempo real de todas las entradas de datos existentes y se ejecutará ininterrumpidamente. • Temperatura exterior, obtenida por un sensor que está instalado en un microcontolador Arduino, y que posteriormente he subido a una plataforma de internet de las cosas llamada Midgar. • Humedad exterior, obtenida de una plataforma de internet de las cosas llamada ThingSpeak • Temperatura interior, obtenida del sensor de temperatura localizado en el microordenador Raspberry desde donde se ejecuta el programa.

Tabla 8 Caso de uso Ejecutar valores tiempo real

Nombre del Caso de Uso
Ejecutar Array de Valores de tamaño introducido por usuario.
Descripción
El usuario ejecutará el programa sin conexión, en este caso se simularán los valores rellenando un Array cuyo tamaño será dado por el usuario. En estos Arrays se crearán datos semialeatorios que oscilan entre valores reales.

Tabla 9 Caso de uso Ejecutar Array de valores tamaño introducido por usuario

Nombre del Caso de Uso
Ejecutar Array de valores semialeatorios
Descripción
El prototipo creará por defeco tres array de datos aleatorios que conllevarán una relación entre cada posición del Array simulado incrementos de temperatura y humedad.

Tabla 10 Caso de uso Ejecutar Array de valores semialeatorios

Nombre del Caso de Uso
Ejecutar valores unitarios
Descripción
El usuario ejecutará el programa sin conexión y se le pedirá que agregue los datos de temperatura y humedad a su valor, para así comprobar el funcionamiento del prototipo con datos unitarios y más concretos.

Tabla 11 Caso de uso ejecutar valores unitarios

Nombre del Caso de Uso
Ejecutar hilo con valores aleatorios
Descripción
El usuario ejecutará el programa sin conexión, y creará datos simulados aleatorios cuyos incrementos y decrecimientos simulan el funcionamiento real de cambios de temperatura y de humedad, es decir, oscilaciones de entre ±2 grados centígrados de temperatura y ±5 puntos de porcentaje en la humedad.

Tabla 12 Caso de uso Ejecutar hilo con valores aleatorios.

5.3 Identificación de los Subsistemas en la Fase de Análisis

El prototipo se ha divido en los siguientes subsistemas:

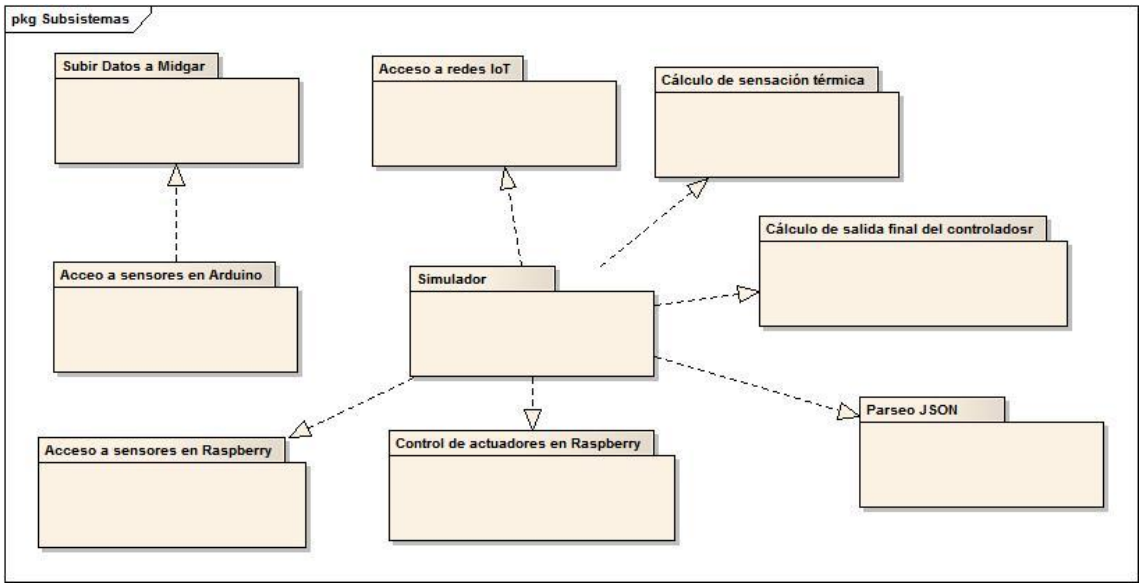


Ilustración 26 Subsistemas

5.3.1 Descripción de los Subsistemas

A continuación se detallarán los distintos subsistemas en los que se divide el prototipo.

Simulador:

- Realiza la conexión con el resto de servicios para la obtención de datos.
- Realiza las operaciones con lógica difusa y muestra los resultados.
- Controla el prototipo ofreciendo una señal final.

Acceso a redes IoT:

- Conexión con la red de internet de las cosas MIDGAR.
- Conexión con la red de internet de las cosas llamada ThingSpeak.

Cálculo de sensación térmica:

- Obtención de temperatura exterior.
- Obtención de humedad exterior.
- Creación de reglas difusas para la sensación térmica

- Cálculo mediante lógica difusa de sensación térmica.

Cálculo de salida final del controlador:

- Obtención de temperatura interior.
- Obtención de sensación térmica.
- Creación de reglas difusas para el cálculo del valor final.
- Cálculo mediante lógica difusa del valor de la salida final.

Parseo de JSON:

- Obtención de los datos desde ThingSpeak en formato JSON.
- Parseo de los datos JSON obtenidos para poder procesarlos ellos.

Control de actuadores de Raspberry:

- Acceso a los controladores de Raspberry.
- Control del uso actuadores instalados en Raspberry, como pueden ser Led's's.

Acceso a sensores en Raspberry;

- Acceso a sensores en Raspberry.
- Control de sensores en Raspberry.
- Obtención de datos de sensores.

Acceso a sensores en Arduino:

- Acceso a sensores en Arduino.
- Control de sensores en Arduino.
- Obtención de datos de sensores.

Subir datos a Midgar:

- Obtención de datos.
- Procesamiento de datos.
- Envío de datos a plataforma de datos MIDGAR.

5.3.2 Descripción de los Interfaces entre Subsistemas

La comunicación entre los diferentes sistemas se lleva a cabo de dos maneras. Una de ellas es local, mediante la llamada a los métodos que existen entre distintas clases.

Po otro lado existe comunicación por red de algunos subsistemas de manera que por ejemplo, para el acceso a datos en las redes de internet de las cosas se hará mediante peticiones HTTP.

Existe un subsistema que no está conectado al resto de los subsistemas, es decir al conjunto de subsistemas principales.

Estos dos subsistemas conectados entre sí y a la vez separados del conjunto principal son:

- Acceso a sensores en Arduino
- Subir Datos a Midgar

Estos dos subsistemas se han desarrollado con la función de lograr acceder a los sensores de temperatura instalados en Arduino, para ellos se ha desarrollado un programa en C que permita leer de otro sensor de temperatura TMP36 instalado en una placa Arduino.

Una vez leídos los datos con el programa instalado en Arduino, acoplaremos nuestro microcontolador a un PC y desarrollaremos una pequeña aplicación en Java que recoja el dato del sensor de temperatura, parsearlo para que se pueda subir correctamente a la plataforma de Internet de las cosas Midgar y así posteriormente podamos consumirlo desde la aplicación principal.

5.4 Diagrama de Clases Preliminar del Análisis

5.4.1 Diagrama de Clases

El siguiente diagrama de clases, es el que representa de manera inicial los distintos subsistemas en los que se divide el prototipo. Este diagrama puede no albergar ninguna similitud con el diagrama final, puesto a lo largo del desarrollo del prototipo y análisis de los módulos se hayan eliminado, modificado o creado nuevas las clases de las que a continuación aparecen.

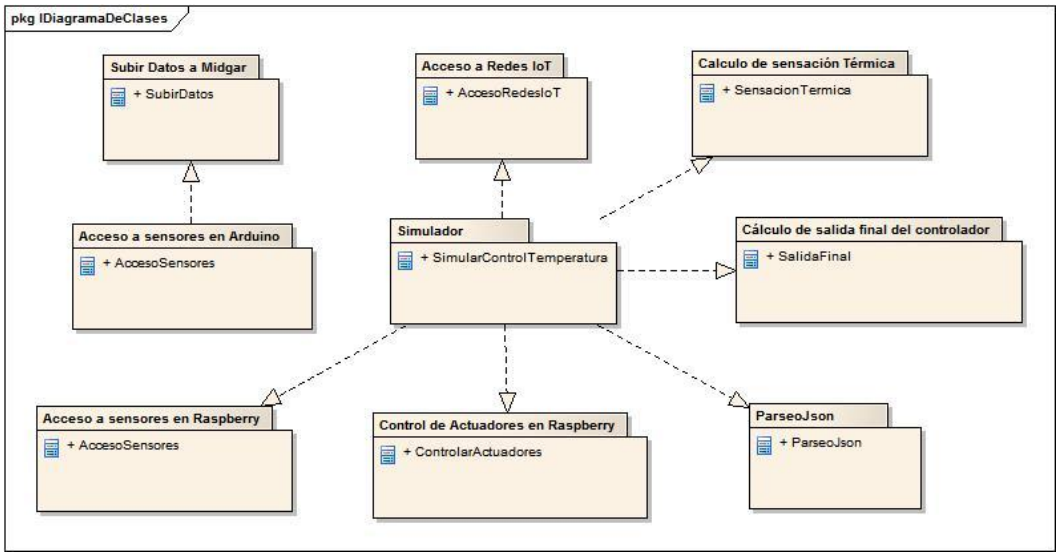


Ilustración 27 Diagrama de clases inicial

5.4.2 Descripción de las Clases

5.4.2.1 *Acceso a redes IoT*

Nombre de la Clase
AccesoRedesIoT
Descripción
Acceso a las redes de Internet de las cosas.
Responsabilidades
Se encargará de acceder a los datos que se encuentran en las redes de Internet de las cosas que hemos escogido para consumir datos.
Métodos Propuestos
HttpGetClassThingSpeak: Accede a la red IoT ThingSpeak. **HttpGetClassMidgar**: Accede a la red IoT Midgar.

Tabla 13 Clase AccesoRedesIot

5.4.2.2 *Cálculo de sensación térmica*

Nombre de la Clase
SensacionTermica
Descripción
Calculará la sensación térmica exterior.
Responsabilidades
Se encargará de realizar el cálculo de sensación térmica exterior mediante lógica difusa.
Métodos Propuestos
GetTempExt: Obtiene la temperatura exterior. **GetHumExt**: Obtiene la humedad exterior. **LoadFuzzyRules**: Cargará las reglas de lógica difusas creadas. **CalculoSensacionTermicaDifusa**: Calculará la sensación térmica exterior mediante lógica difusa.

Tabla 14 Clase SensacionTermica

5.4.2.3 *Cálculo de salida final del controlador.*

Nombre de la Clase
SalidaFinal
Descripción
Calculará la salida final del controlador.
Responsabilidades
Se encargará de realizar el cálculo de la salida final que dará orden al controlador de temperatura mediante lógica difusa.
Métodos Propuestos
GetSensacionTermica: Obtiene la sensación térmica. **GetTemInt**: Obtiene la temperatura interior.

LoadFuzzyRules: Cargará las reglas de lógica difusas creadas. **CalculoDeSalidaFinal**: Calculará la salida final que dará la orden al controlador de temperatura mediante lógica difusa.

Tabla 15 Clase SalidaFinal

5.4.2.4 *ParseoJson*

Nombre de la Clase
PaseoJson
Descripción
Parseará los datos obtenidos en formato Json
Responsabilidades
Se encargará del parseo de datos Json obtenidos de las redes de Internet de las cosas para poder tratarlo posteriormente.
Métodos Propuestos
ParsThingSpeak: Parseo de los datos obtenidos desde ThingSpeak. **ParsMidgar**: Parseo de los datos obtenidos desde Midgar.

Tabla 16 Clase PaseoJson

5.4.2.5 *Control de actuadores en Raspberry*

Nombre de la Clase
ControlarActuadores
Descripción
Controlará los actuadores instalados en una Raspberry.
Responsabilidades
Se encargará del control de los actuadores instalados en una Raspberry con los que posteriormente se representará la salida final, en este caso, a través de Led's's.
Métodos Propuestos
ActivarLeds: Activación de los Led's correspondientes según la salida recogida.

Tabla 17 Clase ControlarActuadores

5.4.2.6 *Acceso a sensores en Raspberry*

Nombre de la Clase
AccesoSensores.
Descripción
Accederá a los sensores instalados en una Raspberry.
Responsabilidades
Se encargará de acceder al sensor de temperatura instalado en una Raspberry, obtener los datos requeridos y trabajar con ellos.
Métodos Propuestos
AccessSensor: Accederá al sensor de temperatura para obtener el dato de temperatura local.

Tabla 18 Clase AccesoSensores

5.4.2.7 *Simulador*

Nombre de la Clase

SimularControlTemperatura.
Descripción
Simulará el control de temperatura con los datos obtenidos.
Responsabilidades
Se encargará de calcular y simular el control de temperatura después de tratar los datos obtenidos a través de los distintos sensores mediante lógica difusa.
Métodos Propuestos
GetDatos: Obtendrá los datos requeridos para simular el controlador. **SimularControlador:** Activará el prototipo y pondrá en marcha su funcionamiento.

Tabla 19 Clase SimularControlTemperatura

5.4.2.8 *Acceso a sensores en Arduino*

Nombre de la Clase
Accesosensores.
Descripción
Accederá a los sensores instalados en un Arduino.
Responsabilidades
Se encargará de acceder al sensor de temperatura instalado en un Arduino, obtener los datos requeridos y trabajar con ellos.
Métodos Propuestos
AccessSensor: Accederá al sensor de temperatura para obtener el dato de temperatura exterior.

Tabla 20 Clase Accesosensores

5.4.2.9 *Subir datos a Midgar*

Nombre de la Clase
SubirDatos
Descripción
Subirá los datos a la red IoT Midgar.
Responsabilidades
Se encargará de subir los datos obtenidos del sensor de temperatura que está instalado en un Arduino a la red Social de Internet de las cosas Midgar, para que después podamos consumir esos datos desde el prototipo.
Métodos Propuestos
ObtenerDatos: Obtendrá los datos del sensor de temperatura instalado en un Arduino. ParsearDatos: Se parsearán los datos en el formato que Midgar para servirlos posteriormente. **SubirDatos:** Subirá los datos a la red de Internet de las cosas Midgar.

Tabla 21 Clase SubirDatos

5.5 Análisis de Casos de Uso y Escenarios

5.5.1 Caso de uso 1: Iniciar prototipo

Escenario 1. Iniciar prototipo	
Precondiciones	Usuario tiene que tener algún tipo de conocimiento sobre Internet de las cosas o sensores. El prototipo tiene que estar correctamente instalado.
Poscondiciones	El usuario pondrá en marcha el prototipo.
Actores	Usuario no identificado
Descripción	El usuario pondrá en marcha el prototipo para escoger el tipo de ejecución que desea realizar posteriormente.
Excepciones	
Notas	
Corresponde al requisito	R1

Tabla 22 Escenario 1

5.5.2 Caso de uso 2. Ejecutar valores en tiempo real

Escenario 2. Ejecutar valores tiempo real	
Precondiciones	El prototipo debe de estar en marcha. El prototipo debe de estar conectado a Internet y los servicios a los que accedan deben de estar en funcionamiento.
Poscondiciones	El prototipo mostrará la salida del controlador, correspondiendo a la situación en tiempo real de los datos obtenidos. El prototipo no se detendrá a menos que el usuario lo haga o exista algún otro problema de conexión.
Actores	Usuario no identificado
Descripción	El usuario ejecutará el programa de modo que se obtendrán valores en tiempo real de todas las entradas de datos existentes y se ejecutará ininterrumpidamente. Se consumirán los siguientes datos. • Temperatura exterior, obtenida por un sensor que está instalado en un microcontolador Arduino, y que posteriormente he subido a una plataforma de internet de las cosas llamada Midgar. • Humedad exterior, obtenida de una plataforma de internet de las cosas llamada ThingSpeak • Temperatura interior, obtenida del sensor de temperatura localizado en el microordenador Raspberry desde donde se ejecuta el programa.
Excepciones	• No existe conexión a internet. • Algunos de las plataformas de internet de las cosas no estén en funcionamiento.
Corresponde al	R1.1

requisito	

Tabla 23 Escenario 2

5.5.3 Caso de uso 3. Ejecutar valores unitarios.

Escenario 3. Ejecutar Array de valores de tamaño introducido por usuario	
Precondiciones	El prototipo debe de estar en marcha. El prototipo debe de estar conectado a Internet y los servicios a los que accedan deben de estar en funcionamiento.
Poscondiciones	El prototipo mostrará la salida del controlador, correspondiendo a la situación en tiempo real de los datos obtenidos.
Actores	Usuario no identificado
Descripción	El usuario ejecutará el programa sin conexión, en este caso se simularán los valores de manera que el usuario introducirá los valores por pantalla según se le vayan pidiendo.
Excepciones	
Corresponde al requisito	R1.2

Tabla 24 Escenario 3

5.5.4 Caso de uso 4. Ejecutar Array de valores de tamaño introducido por usuario.

Escenario 4. Ejecutar Array de valores de tamaño introducido por usuario	
Precondiciones	El prototipo debe de estar en marcha.
Poscondiciones	El prototipo mostrará la salida del controlador, correspondiendo a valores creados aleatoriamente y guardados en un Array.
Actores	Usuario no identificado
Descripción	El usuario ejecutará el programa sin conexión, en este caso se simularán los valores rellenando un Array cuyo tamaño será dado por el usuario. En estos Arrays se crearán datos semialeatorios que están comprendidos en rangos de los que se pueden obtener con valores reales.
Excepciones	
Corresponde al requisito	R1.3

Tabla 25 Escenario 4

5.5.5 Caso de uso 5. Ejecutar Array de valores semialeatorios

Escenario 5. Ejecutar array de valores semialeatorios	
Precondiciones	El prototipo debe de estar en marcha.
Poscondiciones	El prototipo mostrará la salida del controlador, correspondiendo a la situación en tiempo real de los datos obtenidos.
Actores	Usuario no identificado

Descripción	EL prototipo se pondrá en marcha y ejecutará datos semialeatorios creados en tres Arrays para cada entrada. Estos datos están pre configurados y se podrá ver una subida de datos de 2 grados y de 5% de humedad en cada posición del Array.
Excepciones	
Corresponde al requisito	R1.4

Tabla 26 Escenario 5

5.5.6 Caso de Uso 6. Ejecutar hilo con valores semialeatorios.

Escenario 6. Ejecutar valores semialeatorios.	
Precondiciones	El prototipo debe de estar en marcha.
Poscondiciones	El prototipo mostrará la salida del controlador, correspondiendo a valores creados aleatoriamente y guardados en varios Arrays.
Actores	Usuario no identificado
Descripción	Se ejecutará el programa sin conexión, se crearán datos que están comprendidos entre rangos de valores reales cuyos incrementos y decrecimientos simulan el funcionamiento real de cambios de temperatura y de humedad, es decir, oscilaciones de entre ±2 grados centígrados de temperatura y ±5 puntos de porcentaje en la humedad.
Excepciones	
Corresponde al requisito	R1.5

Tabla 27 Escenario 6

5.6 Relación Escenarios-Casos de Uso.

A continuación, de manera resumida y más visible se mostrará una tabla resumen que mostrará la relación Escenarios-Casos de uso.

Casos de uso:	1	2	3	4	5	6
Escenarios						
1	x					
2		x				
3			x			
4				x		
5					x	
6						x

Tabla 28 Relación Escenarios-Casos de Uso

5.7 Especificación del Plan de Pruebas

Para la especificación del plan de pruebas se ha tenido principal preocupación en la confección de las reglas de lógica difusa, ya que forman parte importante del cálculo de valores importante para el funcionamiento correcto del prototipo.

Pruebas Unitarias:

Estas pruebas se han llevado a cabo en todos los métodos que realizan operaciones básicas. Entre estos métodos se encuentra los que trabajan con la lógica difusa. Se ha llevado a cabo de manera que se han comprobado las reglas difusas que se encuentran en los archivos .fcl.

Para la realización de estas pruebas se creó una clase asociada a un archivo .fcl externo al proyecto del prototipo donde se probaban las reglas que se iban a utilizar comparando la salida de cada una de las reglas con el resultado esperado.

A continuación se especifican las pruebas que se llevaron a cabo en cada uno de los archivos de lógica difusa.

Estas pruebas unitarias no se corresponden a un caso de uso en concreto, sino que eran fundamentales para el correcto uso de todos los casos ya que son las comprobaciones de las salidas de los cálculos de sensaciones térmicas y de la salida final del prototipo.

Para el cálculo de las sensaciones térmicas se definieron las siguientes variables lingüísticas asociadas a los siguientes rangos de temperaturas y de humedad:

Variable de entrada **temp_arduino** la cual representa la temperatura exterior en grados centígrados.

Término	Valor mínimo	Valor máximo
extremLow	-15°	-5°
veryLow	-7°	3°
Low	0°	15°
Normal	12°	24°
High	22°	30°
VeryHigh	28°	38°
ExtremHigh	35°	40°

Tabla 29 Definición de variable temp_arduino

Variable de entrada **hum_thinkSpeak la cual representa la humedad exterior en porcentaje.**

Término	Valor mínimo	Valor máximo
veryLow	0%	40%
Low	30%	60%
Normal	50%	70%
High	60%	80%
VeryHigh	75%	100%

Tabla 30 Definición de variable hum_thinkSpeak

Variable de salida **ext_cond** que representa la sensación térmica exterior.

Término	Valor mínimo	Valor máximo
extremLow	-15°	-5°
veryLow	-7°	3°
Low	0°	15°
Normal	12°	24°
High	22°	30°
VeryHigh	28°	38°
ExtremHigh	35°	40°

Tabla 31 Definición de variable ext_cond

A continuación se muestran las reglas difusas para el cálculo de sensaciones térmicas exteriores.

Cálculo de sensaciones térmicas.	
Prueba	**Resultado Esperado**
Temp_arduino = extremLow	ext_cond = extrmLow
Hum_thinkSepeak = veryLow	
Prueba	**Resultado Esperado**
Temp_arduino = extremLow	ext_cond = extrmLow
Hum_thinkSepeak = low	
Prueba	**Resultado Esperado**
Temp_arduino = extremLow	ext_cond = extrmLow
Hum_thinkSepeak = normal	
Prueba	**Resultado Esperado**
Temp_arduino = extremLow	ext_cond = veryLow
Hum_thinkSepeak = High	
Prueba	**Resultado Esperado**
Temp_arduino = extremLow	ext_cond = low
Hum_thinkSepeak = veryhigh	
Prueba	**Resultado Esperado**
Temp_arduino = veryLow	ext_cond = veryLow
Hum_thinkSepeak = veryLow	
Prueba	**Resultado Esperado**
Temp_arduino = veryLow	ext_cond = veryLow
Hum_thinkSepeak = Low	
Prueba	**Resultado Esperado**
Temp_arduino = veryLow	ext_cond = veryLow
Hum_thinkSepeak = normal	
Prueba	**Resultado Esperado**
Temp_arduino = veryLow	ext_cond = Low
Hum_thinkSepeak = high	
Prueba	**Resultado Esperado**
Temp_arduino = veryLow	ext_cond = Low
Hum_thinkSepeak = veryHigh	
Prueba	**Resultado Esperado**
Temp_arduino = Low	ext_cond = veryLow
Hum_thinkSepeak = veryLow	

Prueba	**Resultado Esperado**
Temp_arduino = Low	ext_cond = Low
Hum_thinkSepeak = Low	
Prueba	**Resultado Esperado**
Temp_arduino = Low	ext_cond = Low
Hum_thinkSepeak = normal	
Prueba	**Resultado Esperado**
Temp_arduino = Low	ext_cond = Low
Hum_thinkSepeak = high	
Prueba	**Resultado Esperado**
Temp_arduino = Low	ext_cond = normal
Hum_thinkSepeak = veryHigh	
Prueba	**Resultado Esperado**
Temp_arduino = normal	ext_cond = normal
Hum_thinkSepeak = veryLow	
Prueba	**Resultado Esperado**
Temp_arduino = normal	ext_cond = normal
Hum_thinkSepeak = Low	
Prueba	**Resultado Esperado**
Temp_arduino = normal	ext_cond = normal
Hum_thinkSepeak = normal	
Prueba	**Resultado Esperado**
Temp_arduino = normal	ext_cond = normal
Hum_thinkSepeak = high	
Prueba	**Resultado Esperado**
Temp_arduino = normal	ext_cond = normal
Hum_thinkSepeak = veryHigh	
Prueba	**Resultado Esperado**
Temp_arduino = high	ext_cond = normal
Hum_thinkSepeak = veryLow	
Prueba	**Resultado Esperado**
Temp_arduino = high	ext_cond = high
Hum_thinkSepeak = Low	
Prueba	**Resultado Esperado**
Temp_arduino = high	ext_cond = veryHigh
Hum_thinkSepeak = normal	
Prueba	**Resultado Esperado**
Temp_arduino = high	ext_cond = veryHigh
Hum_thinkSepeak = high	
Prueba	**Resultado Esperado**
Temp_arduino = high	ext_cond = extremHigh
Hum_thinkSepeak = veryHigh	
Prueba	**Resultado Esperado**
Temp_arduino = veryHigh	ext_cond = high
Hum_thinkSepeak = veryLow	
Prueba	**Resultado Esperado**
Temp_arduino = veryHigh	ext_cond = veryHigh
Hum_thinkSepeak = Low	
Prueba	**Resultado Esperado**
Temp_arduino = veryHigh	ext_cond = extremHigh
Hum_thinkSepeak = normal	

Prueba	**Resultado Esperado**
Temp_arduino = veryHigh	ext_cond = extremHigh
Hum_thinkSepeak = high	
Prueba	**Resultado Esperado**
Temp_arduino = veryHigh	ext_cond = extremHigh
Hum_thinkSepeak = veryHigh	
Prueba	**Resultado Esperado**
Temp_arduino = extremHigh	ext_cond = veryHigh
Hum_thinkSepeak = veryLow	
Prueba	**Resultado Esperado**
Temp_arduino = extremHigh	ext_cond = extremHigh
Hum_thinkSepeak = Low	
Prueba	**Resultado Esperado**
Temp_arduino = extremHigh	ext_cond = extremHigh
Hum_thinkSepeak = normal	
Prueba	**Resultado Esperado**
Temp_arduino = extremHigh	ext_cond = extremHigh
Hum_thinkSepeak = high	
Prueba	**Resultado Esperado**
Temp_arduino = extremHigh	ext_cond = extremHigh
Hum_thinkSepeak = veryHigh	

Tabla 32 Reglas difusa cálculo sensaciones térmicas

Para el cálculo de la salida final del prototipo se definieron las siguientes variables lingüísticas asociadas a los siguientes rangos de temperaturas:

Variable de entrada **ext_cond** la cual representa la sensación térmica exterior.

Término	Valor mínimo	Valor máximo
extremLow	-15°	-5°
veryLow	-7°	3°
Low	0°	15°
Normal	12°	24°
High	22°	30°
VeryHigh	28°	38°
ExtremHigh	35°	50°

Tabla 33 Definición de variable ext_cond

Variable de entrada **temp_local** que representa la temperatura interior.

Término	Valor mínimo	Valor máximo
extremLow	-15°	-5°
veryLow	-7°	3°
Low	0°	15°
Normal	12°	24°
High	22°	30°
VeryHigh	28°	38°
ExtremHigh	35°	50°

Tabla 34 Definición de variable temp_local

Variable de salida **signal** que representa la salida final del prototipo.

Término	Valor mínimo	Valor máximo
On2	-15°	15°
On	10°	20°
Stable	18°	22°
Off	20°	30°
Off2	25°	50°

Tabla 35 Definición de variable signal

A continuación se muestran las reglas difusas para el cálculo de la salida final.

Cálculo de salida final.	
Prueba	**Resultado Esperado**
Ext_cond = extremLow	signal = on2
Temp_local = extremLow	
Prueba	**Resultado Esperado**
Ext_cond = extremLow	signal = on2
Temp_local = veryLow	
Prueba	**Resultado Esperado**
Ext_cond = extremLow	signal = on2
Temp_local = low	
Prueba	**Resultado Esperado**
Ext_cond = extremLow	signal = on
Temp_local = normal	
Prueba	**Resultado Esperado**
Ext_cond = extremLow	signal = stable
Temp_local = High	
Prueba	**Resultado Esperado**
Ext_cond = extremLow	signal = off
Temp_local = veryhigh	
Prueba	**Resultado Esperado**
Ext_cond = extremLow	signal = off2
Temp_local = extremHigh	
Prueba	**Resultado Esperado**
Ext_cond = veryLow	signal = on2
Temp_local = extremLow	
Prueba	**Resultado Esperado**
Ext_cond = veryLow	signal = on2
Temp_local = veryLow	
Prueba	**Resultado Esperado**
Ext_cond = veryLow	signal = on2
Temp_local = low	
Prueba	**Resultado Esperado**
Ext_cond = veryLow	signal = on
Temp_local = normal	
Prueba	**Resultado Esperado**
Ext_cond = veryLow	signal = stable
Temp_local = High	

Prueba	**Resultado Esperado**
Ext_cond = veryLow	signal = off
Temp_local = veryhigh	
Prueba	**Resultado Esperado**
Ext_cond = veryLow	signal = off2
Temp_local = extremHigh	
Prueba	**Resultado Esperado**
Ext_cond = low	signal = on2
Temp_local = extremLow	
Prueba	**Resultado Esperado**
Ext_cond = low	signal = on2
Temp_local = veryLow	
Prueba	**Resultado Esperado**
Ext_cond = low	signal = on2
Temp_local = low	
Prueba	**Resultado Esperado**
Ext_cond = low	signal = on
Temp_local = normal	
Prueba	**Resultado Esperado**
Ext_cond = low	signal = stable
Temp_local = High	
Prueba	**Resultado Esperado**
Ext_cond = low	signal = off
Temp_local = veryhigh	
Prueba	**Resultado Esperado**
Ext_cond = low	signal = off2
Temp_local = extremHigh	
Prueba	**Resultado Esperado**
Ext_cond = normal	signal = on2
Temp_local = extremLow	
Prueba	**Resultado Esperado**
Ext_cond = normal	signal = on2
Temp_local = veryLow	
Prueba	**Resultado Esperado**
Ext_cond = normal	signal = on
Temp_local = low	
Prueba	**Resultado Esperado**
Ext_cond = normal	signal = stable
Temp_local = normal	
Prueba	**Resultado Esperado**
Ext_cond = normal	signal = stable
Temp_local = High	
Prueba	**Resultado Esperado**
Ext_cond = normal	signal = off
Temp_local = veryhigh	
Prueba	**Resultado Esperado**
Ext_cond = normal	signal = off2
Temp_local = extremHigh	
Prueba	**Resultado Esperado**
Ext_cond = high	signal = on2
Temp_local = extremLow	

Prueba	**Resultado Esperado**
Ext_cond = high	signal = on2
Temp_local = veryLow	
Prueba	**Resultado Esperado**
Ext_cond = high	signal = on
Temp_local = low	
Prueba	**Resultado Esperado**
Ext_cond = high	signal = stable
Temp_local = normal	
Prueba	**Resultado Esperado**
Ext_cond = high	signal = off
Temp_local = High	
Prueba	**Resultado Esperado**
Ext_cond = high	signal = off2
Temp_local = veryhigh	
Prueba	**Resultado Esperado**
Ext_cond = high	signal = off2
Temp_local = extremHigh	
Prueba	**Resultado Esperado**
Ext_cond = veryHigh	signal = on2
Temp_local = extremLow	
Prueba	**Resultado Esperado**
Ext_cond = veryHigh	signal = on2
Temp_local = veryLow	
Prueba	**Resultado Esperado**
Ext_cond = veryHigh	signal = on
Temp_local = low	
Prueba	**Resultado Esperado**
Ext_cond = veryHigh	signal = stable
Temp_local = normal	
Prueba	**Resultado Esperado**
Ext_cond = veryHigh	signal = off
Temp_local = High	
Prueba	**Resultado Esperado**
Ext_cond = veryHigh	signal = off2
Temp_local = veryhigh	
Prueba	**Resultado Esperado**
Ext_cond = veryHigh	signal = off2
Temp_local = extremHigh	
Prueba	**Resultado Esperado**
Ext_cond = extremHigh	signal = on2
Temp_local = extremLow	
Prueba	**Resultado Esperado**
Ext_cond = extremHigh	signal = on2
Temp_local = veryLow	
Prueba	**Resultado Esperado**
Ext_cond = extremHigh	signal = on
Temp_local = low	
Prueba	**Resultado Esperado**
Ext_cond = extremHigh	signal = off
Temp_local = normal	

Prueba	**Resultado Esperado**
Ext_cond = extremHigh	signal = off2
Temp_local = High	
Prueba	**Resultado Esperado**
Ext_cond = extremHigh	signal = off2
Temp_local = veryhigh	
Prueba	**Resultado Esperado**
Ext_cond = extremHigh	signal = off2
Temp_local = extremHigh	

Tabla 36 Reglas difusas para la salida final

Capítulo 6. Diseño del Sistema

6.1 Arquitectura del Sistema

6.1.1 Diagramas de Paquetes

En el siguiente diagrama de paquetes se podrá observar la estructura lógica de cómo se reparten las clases que no coindice con la distribución física, ya que algunas se encuentran en la misma carpeta debido a tratarse de pocas clases.

Este diagrama es el surgido al final de la aplicación donde cada subsistema ha coincidido con su paquete correspondiente.

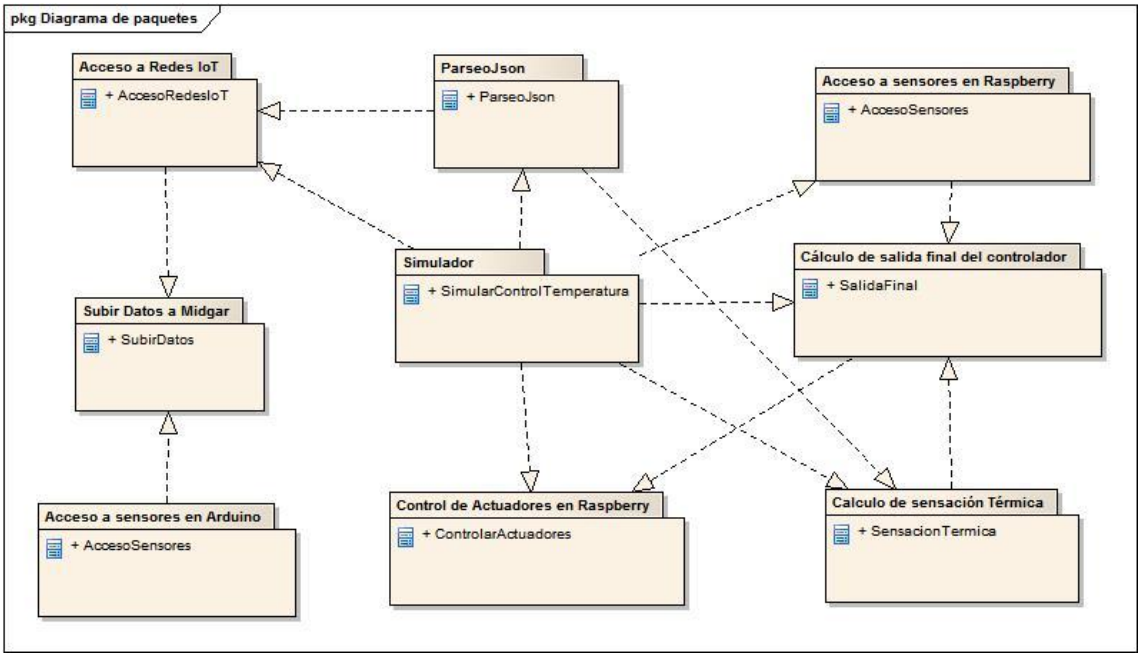


Ilustración 28 Diagrama de paquetes

6.1.2 Diagramas de Componentes

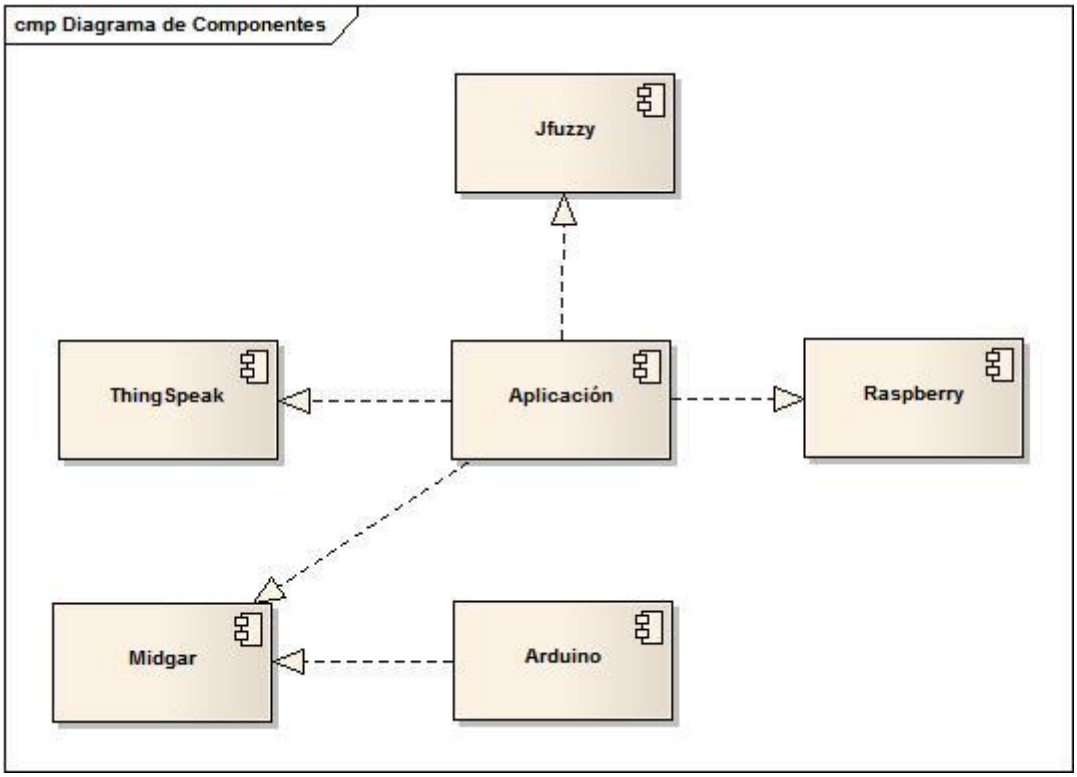


Ilustración 29 Diagrama de componentes

6.1.3 Diagramas de Despliegue

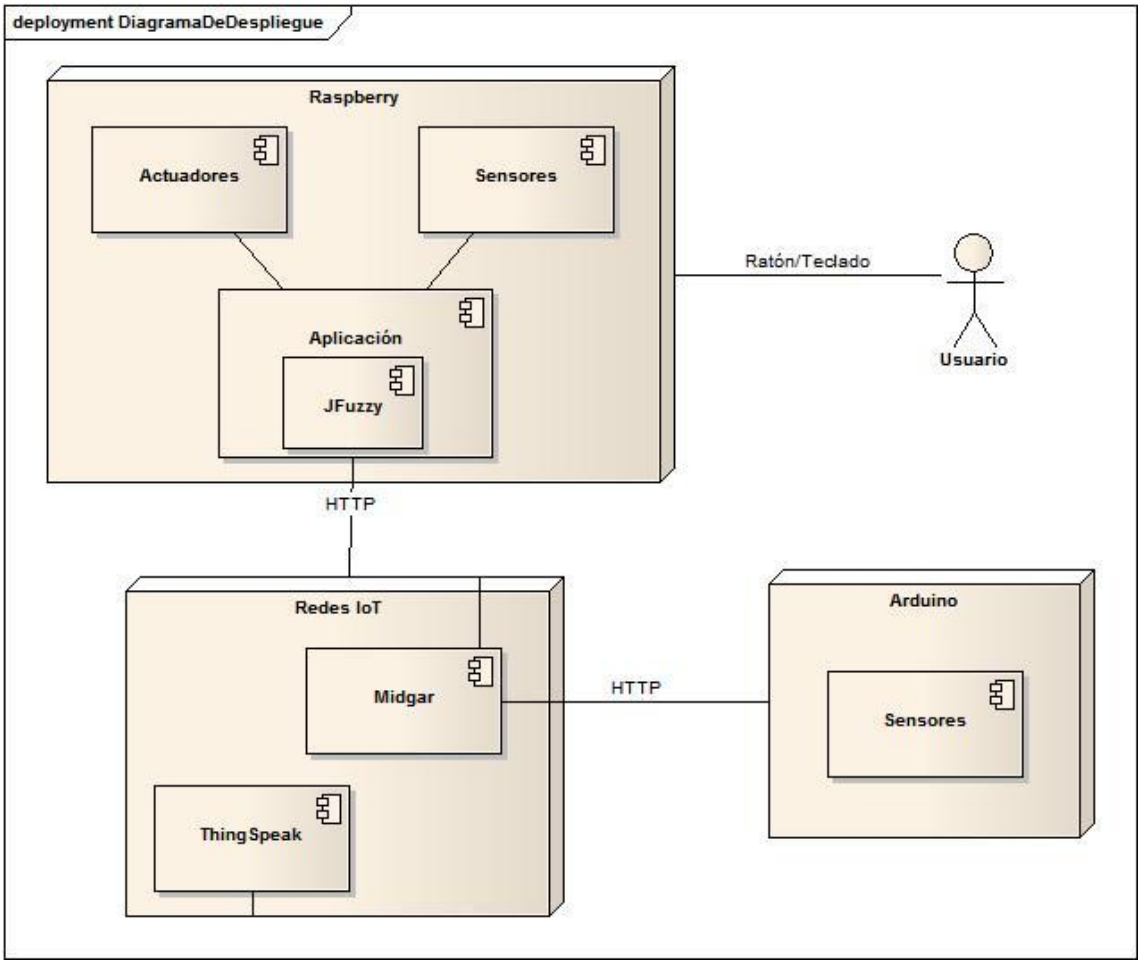


Ilustración 30 Diagrama de despliegue

6.1.3.1 *Raspberry*

La aplicación del prototipo estará instalada en un mini-ordenador en este caso será una Raspberry Pi 2 Model B. Para el funcionamiento correcto del prototipo necesitaremos que tenga instalado un entorno de ejecución Java, puesto que la aplicación está desarrollada completamente en Java, así como una conexión a internet ya que se accederá a otros servicios a través de peticiones HTTP.

La aplicación instalada en la Raspberry se encargará de recoger la información de los sensores que tiene asociados así mismo.

Consumirá datos de las redes de internet de las cosas mediante peticiones HTTP.

Tratará los datos obtenidos a través de las redes IoT y de los sensores, utilizará lógica difusa para los operaciones correspondientes.

Controlará los actuadores asociados al propio microordenador para representar las salidas obtenidas.

6.1.3.2 *Redes IoT*

Principalmente se usarán dos redes de internet de las coas.

- ThingSpeak: Previamente se ha escogido un canal que satisfaga las necesidades de nuestro prototipo. En este caso el canal nos ofrece datos en grados centígrados de un sensor de temperatura.
- Midgar: Consumiremos datos de la red Midgar, datos que provienen de un sensor de temperatura instalado en un microcontolador Arduino y que se ha programado previamente para que podamos consumir los datos que se están ofreciendo.

6.1.3.3 *Arduino*

Microcontolador encargado de recibir datos de un sensor de temperatura instalado y de subirlos a la red de Internet de las cosas Midgar en formato JSON.

6.1.3.4 *Usuario*

Le usuario solo podrá poner en marcha el prototipo desde la Raspberry donde están asociados la configuración e instalación de los sensores y controladores. Se podrá iniciar a través de la ejecución de un programa desarrollado en Java.

6.2 Diseño de Clases

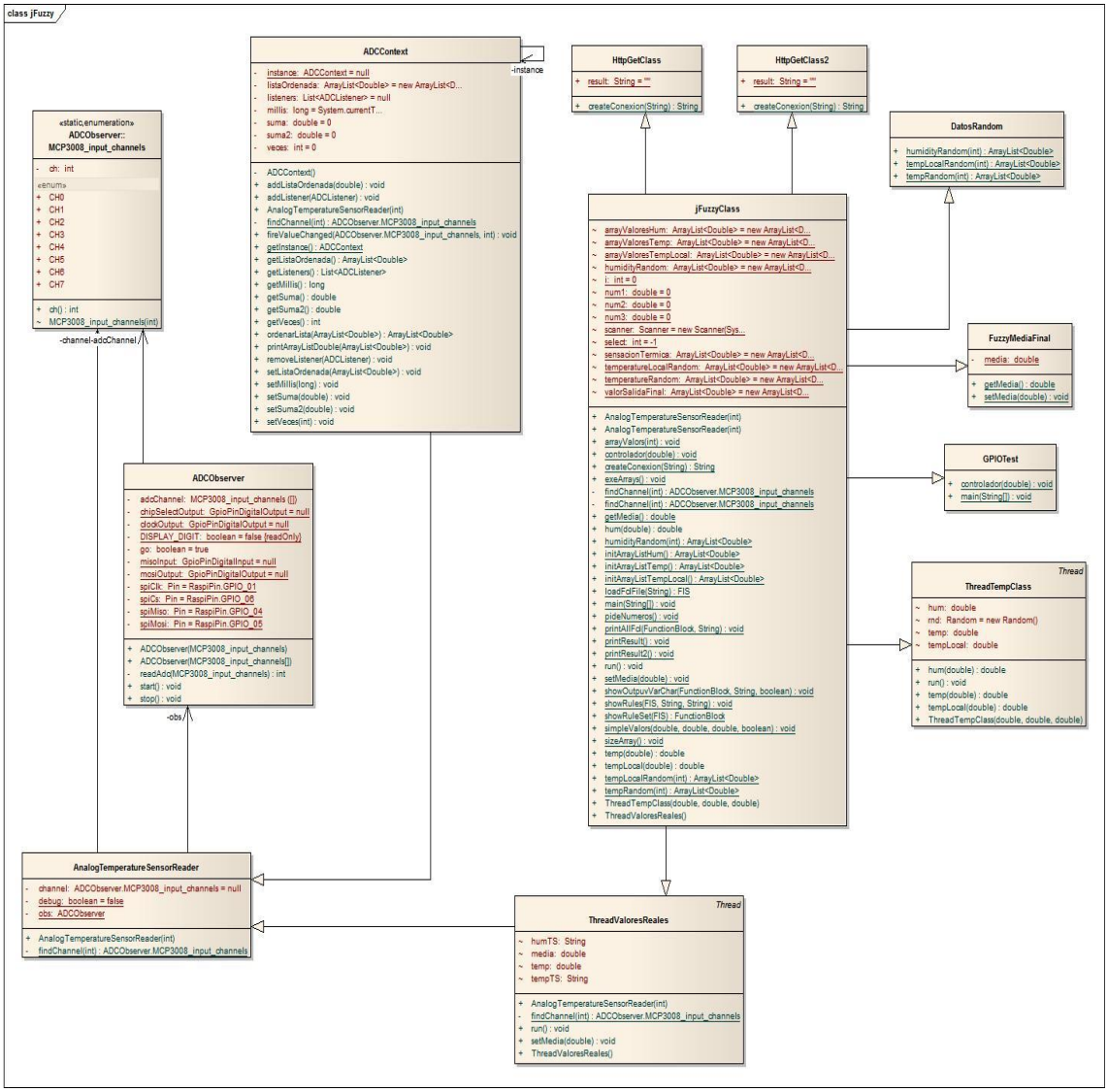


Ilustración 31 Diagrama de clases final

6.3 Diagramas de Interacción y Estados

Se mostrarán a continuación dos diagramas de interacción que resumen todos los casos de uso que hemos definido. Todos ellos estarán iniciados por el usuario no identificado y la diferencia fundamental entre ambos diagramas será si el prototipo está conectado a internet y trabaja con datos reales o si se trata de una simulación de datos semialeatorios.

6.3.1 Caso de Uso 1, 2.

El siguiente diagrama de interacciones representa los casos de uso que obtienen los datos en tiempo real.

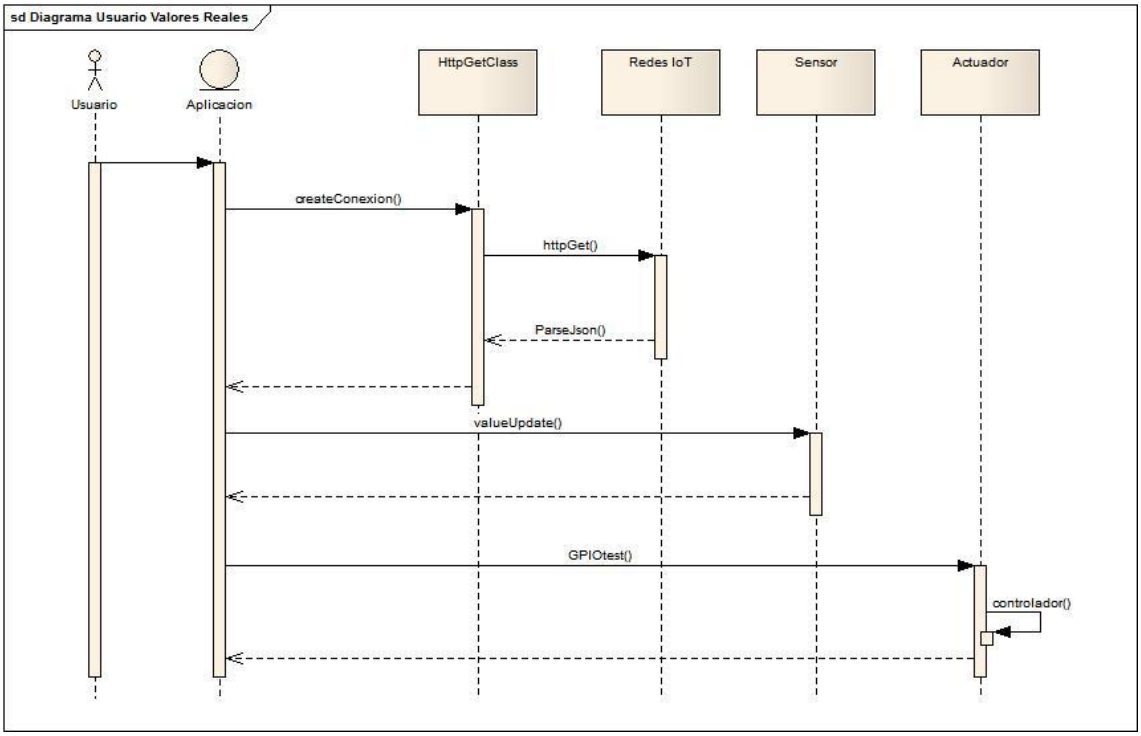


Ilustración 32 Diagrama de Internación 1

6.3.2 Casos de uso 1, 3, 4, 5, 6.

A continuación se mostrará el diagrama de iteraciones que mostrará los casos de uso que generen datos no reales, es decir datos semialeatorios o datos introducidos por el usuario.

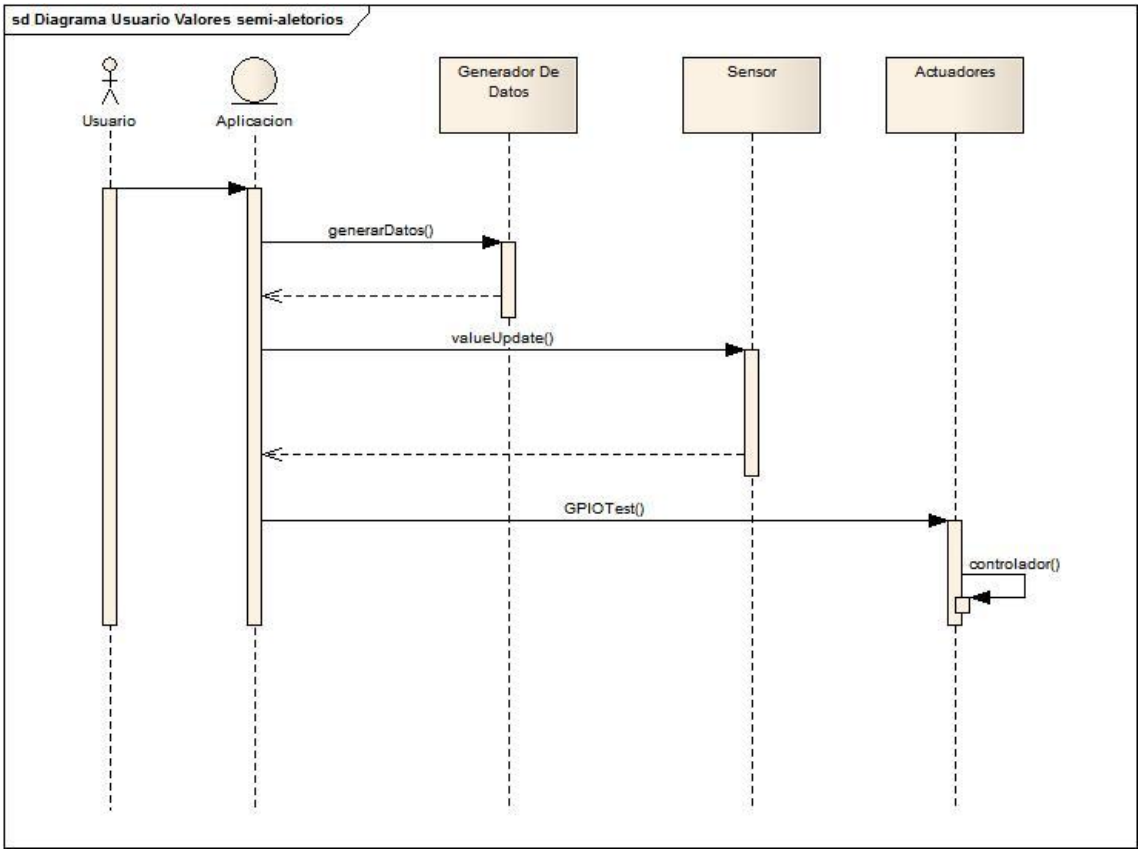


Ilustración 33 Diagrama de Interacción 2

6.4 Diagramas de Actividades

6.4.1 Diagrama de actividades con valores en tiempo real.

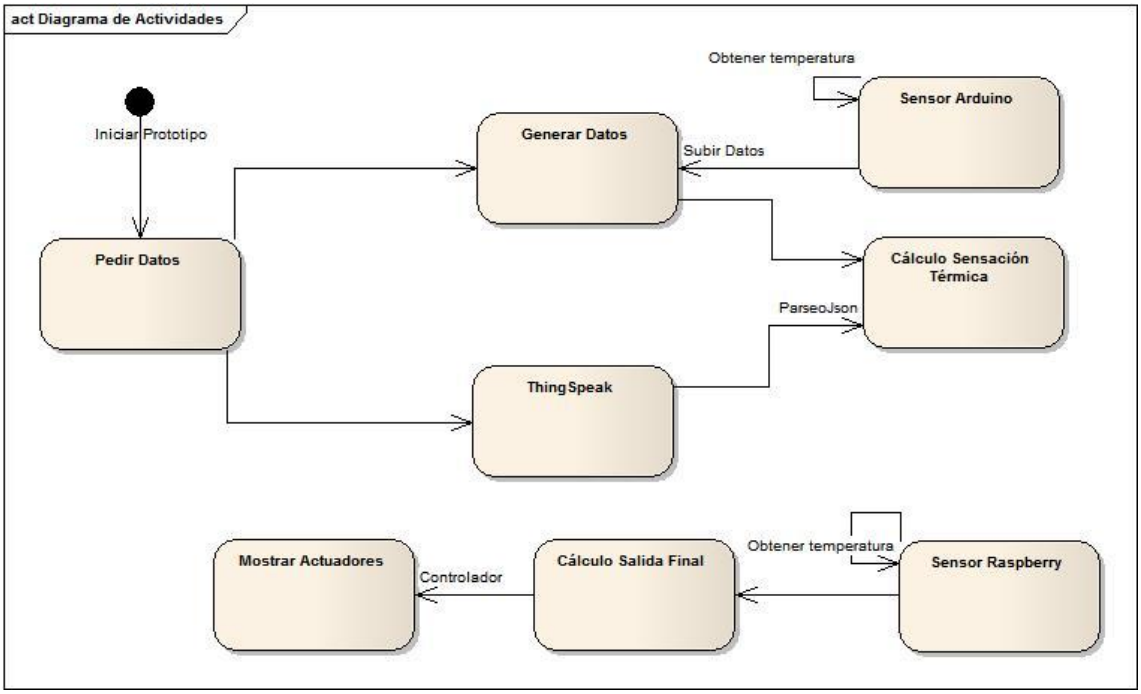


Ilustración 34 Diagrama de actividades con valores en tiempo real

6.4.2 Diagrama de actividades con valores generados semialeatorios.

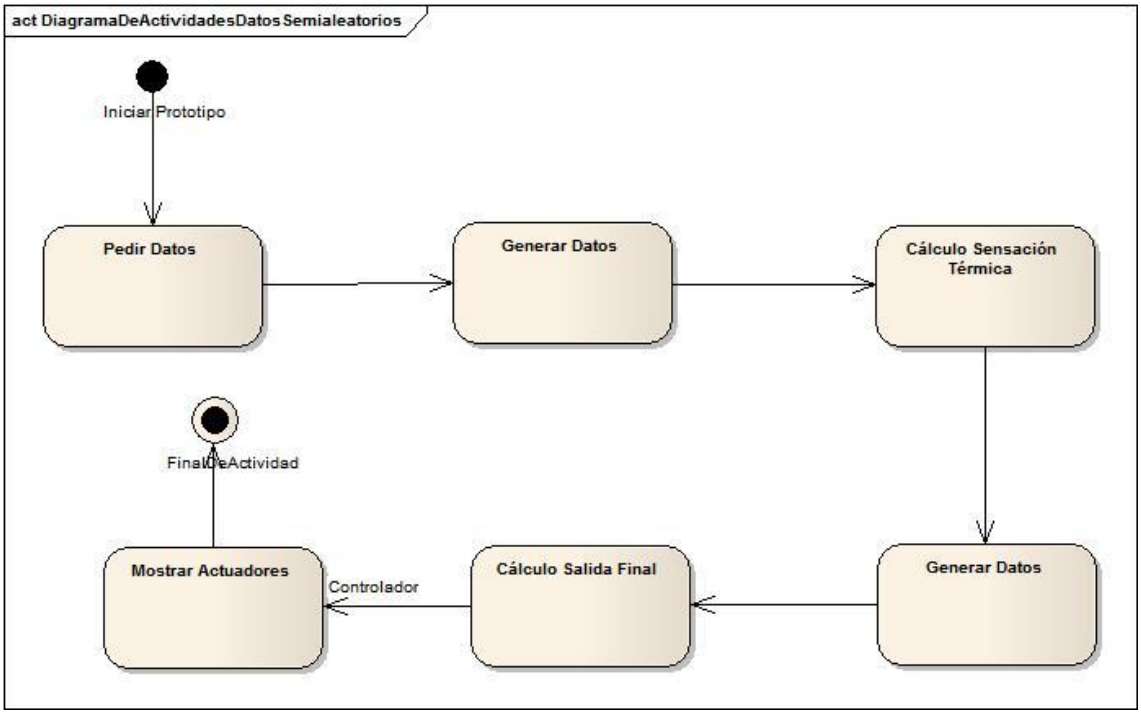


Ilustración 35 Diagrama de actividades con valores aleatorios

6.5 Especificación Técnica del Plan de Pruebas

Antes de describir las pruebas, se va a hacer una especificación de la máquina desde donde se va a llevar a cabo la ejecución de las pruebas del prototipo y la ejecución del programa.

Desde un inicio se ha trabajado en un microordenador Raspberry Pi 2 model B cuyas características se muestran a continuación.

Arquitectura del núcleo	**Quad-core ARM Cortex-A7.**
CPU	900MHz.
GPU	Dual Core VideoCore IV® Multimedia Co-Processor. Provee Open GL ES 2.0, OpenVG acelerado por hardware, y decodificación de alto perfil 1080p30 H.264. Puede con 1Gpixel/s, 1.5Gtexel/s o 24GFLOPs con filtrado de texturas e infraestructura DMA.
Memoria	1Gb LPDDR2
USB	4
Entradas de vídeo	Conector [MIPI] CSI
Salidas de vídeo	Conector Jack (PAL y NTSC), HDMI (rev1.3 y 1.4), interfaz DSI para panel LCD.
Salidas de audio	Conector Jack de 3.5 mm y HDMI.
Almacenamiento integrado:	MicroSD
Conectividad de red:	10/100 Ethernet (RJ-45).
Conector GPIO	40 pines en dos líneas de 20.
Fuente de alimentación	5v vía Micro USB o GPIO header.
Consumo	800mA/4w.
Tamaño de placa	85 x 56 x 17mm.
Sistemas operativos soportados	Debian (Raspbian), Fedora (Pidora), Arch Linux (Arch Linux ARM), Slackware Linux, RISC OS, Windows 10.

Tabla 37 Características específicas de Raspberry Pi 2 Model B

Todas las pruebas se realizaron a través de este equipo ya que es el que se usará para la funcionamiento del prototipo en su versión final.

Se instaló como sistema operativo uno de los las especificaciones técnicas muestran, Debian (Raspbian).

Sobre este sistema operativo se instaló una versión de desarrollo de Eclipse para poder trabajar con la Raspberry Pi y desde ahí realizar las pruebas correspondientes

6.5.1 Pruebas unitarias.

Como ya se habló en la especificación del plan de pruebas, las pruebas unitarias se realizarán sobre las clases o métodos que realicen operaciones basadas en las reglas difusas creadas.

Las reglas difusas en este caso se encuentran en los archivos .fcl del proyecto. Se trata de dos archivos que controlarán dos secciones importantes del prototipo:

- Cálculo de la sensación térmica exterior.
- Cálculo de salida final del prototipo.

Ambos cálculos se realizarán desde una clase de Java que cargará cada archivo por separado y comprobarán las salidas esperadas dependiendo de las variables de entrada obtenidas en función de las reglas establecidas en los archivos .fcl como ya se mostró en la especificación del plan de pruebas.

En el caso de que las pruebas sean correctas se seguirá trabajando con normalidad, pero en el caso que las pruebas sean incorrectas se deberán revisar las reglas difusas así como las variables lingüísticas y establecer unas nuevas hasta que el resultado sea óptimo.

Al trabajar con lógica difusa, los resultados de los bloques de lógica difusa son números concretos que corresponden a variables lingüísticas, es decir a rangos de valores, por lo que en la comprobación de las pruebas, los resultados no tenían que ser valores concretos, sino estar dentro de un rango de valores, es decir dentro de las variables lingüísticas correctas.

Capítulo 7. Implementación del Sistema

7.1 Estándares y Normas Seguidos

7.1.1 Json

Ilustración 36 Logo JSON

Json (JavaScript Object Notation), es un formato ligero para el intercambio de datos que se está usando como alternativa a XML. Una de las supuestas ventajas sobre XML es que en JSON es mucho más sencillo escribir un analizador sintáctico (parser).

Json se suele emplear en entornos donde la cantidad de datos que se están moviendo entre cliente y servidor son de vital importancia cuando la fuente es de fiar y donde no es importante el no disponer de procesamiento XSLT para manipular datos del cliente.

Este término está muy difundido entre los medios de programación ya que la lista de lenguajes que soporta es muy amplia, ActionScript, C, C++, C#, Java, JavaScript, Perl, PHP, Python entre otros, aunque este término está mal descrito ya que en realidad es solo una parte de la definición del **estándar** ECMA-262 basado en Javascript.

La elección de escoger Json principalmente fue como se ha citado anteriormente, la facilidad que existe a la hora de crear los analizadores sintácticos que fueron necesarios para el desarrollo del prototipo.

7.2 Lenguajes de Programación

7.2.1 Java

Ilustración 37 Logo Java

Se trata de un leguaje de programación de propósito general, concurrente, orientado a objetos que fue diseñado específicamente para tener las menores dependencias de implementación como fuera posible.

Las bases de creación de Java fueron las de poder ejecutar un mismo programa en diversos sistemas operativos, inclusión de soporte para trabajo en red, ejecución de código en sistemas remotos de manera segura y facilidad de uso.

Fue originalmente desarrollado por Sun Microsystems que más adelante fue comprado por Oracle, derivando en gran medida de C y C++ aunque tiene menos utilidades de bajo nivel que cualquiera de ellas.

En este ligar se llevó a cabo la elección de Java como lenguaje de programación por:

- Orientación a objetos
- Flexibilidad y reutilización de código para adaptarlo a nuevos entornos.
- Funcionamiento de cualquier plataforma como Windows, Mac, Raspberry…
- Desarrollo gratuito simplemente descargando el JDK correspondiente
- Fuente abierta de casi todas sus librerías nativas
- Variedad de librerías para utilizar

En nuestro caso fue preciso el uso de algunas librerías concretas para trabajar en el desarrollo del prototipo.

7.2.1.1 *Librerías usadas*

7.2.1.1.1 JFuzzy

Ilustración 38 jFuzzy logo

JFuzzy es una librearía de java que permite implementar un lenguaje de control difuso (FCL), el cual estandariza la programación de sistemas de lógica difusa y reduce su tiempo de programación.

JFuzzy soporta muchas funciones de pertenencia, defucificadores, reglas de agregación y operadores de conexión de reglas. Se pueden crear funciones suficientemente complejas así como sus variables parametrizadas.

7.2.1.1.2 Pi4j

Ilustración 39 Logo Pi4j

Esta librería está enfocada en proporcionar a los programadores el control de las capacidades de entrada y salida de la plataforma Raspberry Pi.

Pi4j implementa un esquema de número de Pin´s abstracto para ayudar a aislar el programa de consumir los cambios de Hardware de las placas de Raspberry Pi.

Pi4j implementa el mismo esquema de número de PIN que el proyecto WiringPi. Los cambios de GPIO PIN entre las versiones de la placa 1 y 2 son un ejemplo perfecto de porque se justifica este esquema de números de pin abstracto.

7.2.1.1.3 HTTP Apache

Ilustración 40 Logo Apache

El protocolo de transferencia de HiperTexto (HTTP) puede ser a día de hoy el protocolo más usado en Internet. Servicios Web, dispositivos de red, el crecimiento del desarrollo de programas en la red, los navegadores web orientados al usuario y el uso de aplicaciones que requieren compatibilidad con HTTP sigue ampliando el papel del protocolo HTTP.

HTTP Apache pueden ser de interés para cualquiera en la construcción de aplicaciones cliente y servidor HTTP como navegadores web, arañas web, bibliotecas de transporte de servicios web, o sistemas que aprovechan o extienden el protocolo HTTP para la comunicación.

7.2.2 C

Lenguaje originalmente desarrollado por Dennis M. Ritchie entre 1969 y 1972 como evolución del lenguaje anterior B.

Se trata de un lenguaje orientado a la implementación de Sistemas Operativos, concretamente Unix. Es un lenguaje de tipos de datos estáticos, poco tipificado, de medio nivel pero con características de bajo nivel. Dispone estructuras típicas de lenguajes de alto nivel pero por otro lado permite un control a muy bajo nivel.

Los compiladores ofrecen extensiones al lenguaje que posibilitan al lenguaje mezclar código en ensamblador con código C o acceder directamente a memoria o a dispositivos periféricos.

7.2.2.1 *Propiedades*

- Lenguaje simple y flexible
- Impide operaciones de tipos sin sentido
- Usa de lenguaje de preprocesado
- Acceso a memoria de bajo nivel mediante uso de punteros
- Conjunto reducido de palabras clave
- Puteros a funciones y variables estáticas

7.2.2.2 *Carencias*

- Recolección de basura nativa
- Soporte para programación orientada a objetos
- Funciones anidadas. (GCC tiene esta característica como extensión)
- Soporte nativo para programación multihilo

7.3 Herramientas y Programas Usados para el Desarrollo

Descripción de todas las herramientas de desarrollo, sistemas adicionales, complementos y otros productos software que necesitemos para la implementación de nuestro sistema.

7.3.1 Eclipse

Ilustración 41 Logo Eclipse

7.3.1.1 *Descripción*

Eclipse es una plataforma de desarrollo diseñada para ser extendida de forma indefinida a través de plugins. Fue concebida desde el principio para ser una plataforma de integración de herramientas de desarrollo. No tiene un lenguaje específico, es un IDE genérico, aunque en la comunidad de desarrolladores de lenguaje Java es muy popular usando el plug-in JDT que viene incluido en la distribución estándar del IDE.

Proporciona herramientas para la gestión de espacios de trabajo, escribir, desplegar, ejecutar y depurar aplicaciones.

7.3.1.2 *Ventajas*

- Perspectivas, editores y vistas: Preconfiguración de ventanas y editores, relacionadas entre sí y que permite trabajar en un determinado entorno de trabajo de forma óptima.
- Gestión de proyectos: Proporciona asistentes y ayudas para la creación de proyectos.
- Depurador de código: Potente depurador de uso fácil que nos ayuda a mejorar nuestro código.
- Extensa colección de plug-ins: Disponibilidad de una gran cantidad de ellos, unos publicados por Eclipse y otros por terceros.

7.3.1.3 *Inconvenientes*

- Mayor consumo de recursos del sistema en mayor o menor medida.

7.3.1.4 *Elección*

En este caso se utilizó la última versión disponible en el momento del inicio del desarrollo, para el desarrollo que se realizó en el PC se utilizó:

- Versión: Luna Service Release 1a (4.4.1).

Por otro lado la plataforma usada en la Raspberry Pi fue la disponible para trabajar en este microordenador:

- Versión 3.8.0

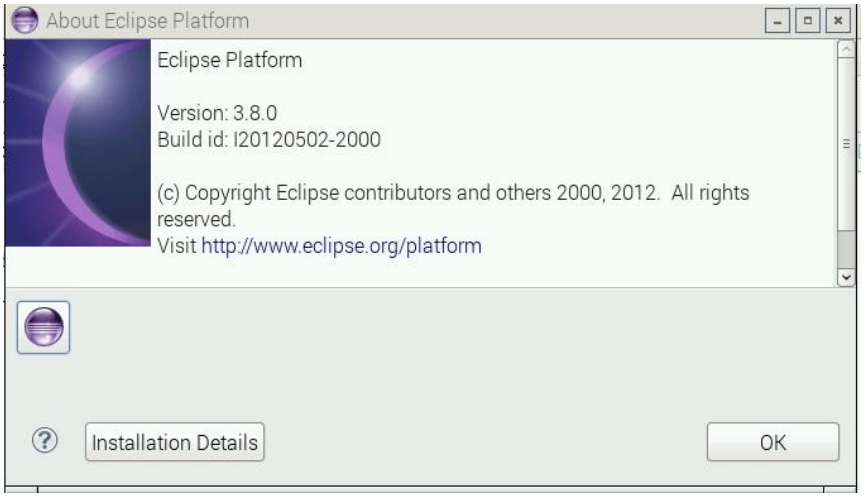


Ilustración 42 Versión Eclipse utilizada en Raspberry Pi.

La elección de esta plataforma fue entre otras debido a que era una plataforma en la que ya se tenía conocimiento del uso de la misma facilitaba el desarrollo.

Se encontraron librerías adecuadas para el trabajo de:

- Lógica Difusa.
- Acceso a los PIN GPIO de Raspberry.
- Peticiones HTTP.
- Parseo de datos.

Otro factor fue que debido a que la aplicación se iba a desarrollar en Java, y Eclipse como se ha dicho anteriormente es una de los principales entornos de programación en este lenguaje.

7.3.2 IDE Arduino

Ilustración 43 Logo Arduino

7.3.2.1 *Descripción*

Se trata de un entorno de desarrollo ofrecido por Arduino orientado a programar aplicaciones para ejecutar en sus placas. Se puede ejecutar en Windows, Mac OS, y Linux.

El entorno está escrito en Java y basado en Processing y otros software de código abierto.

El IDE de Arduino lleva una biblioteca en C++ llamada "Wiring" que hace más fácil escribir los programas.

7.3.2.2 *Ventajas*

- Lenguaje simple basado en C/C++.
- Permite desde un primer contacto estar programando directamente el hardware
- Proyecto Open-Source.
- Comunidad de desarrollo que permite un acceso a referencias, ejemplos y proyectos que sirven de gran ayuda.

7.3.2.3 *Inconvenientes*

- Su uso es específico para las placas Arduino

7.3.2.4 *Elección*

Para el desarrollo de programas sobre la placa Arduino, y en este caso concreto para la lectura de datos del sensor de temperatura era necesario usar un IDE que mostrara las facilidades y el acceso a una gran cantidad de referencias y ejemplos que sirvieron de apoyo para el desarrollo.

Se uso la versión 1.6.5.

7.3.3 Microsoft Office 2007

Ilustración 44 Logo Microsoft Office

7.3.3.1 *Descripción*

Office es una suite ofimática creada por Microsoft para los sistemas operativos de Microsoft Windows y Mac Os. La primera vez que fue lanzada, fue en 1989.

Esta suite cuenta con varios programas que permite el desarrollo de tareas requeridas en la documentación de todo el proyecto.

Ilustración 45 Logo Word 2007

En primer lugar, se usó Microsoft Word como procesador de textos ya que ofrece una gran variedad de características para la maquetación de la documentación. Los índices autogenerados a partir de títulos, variedad de fuentes de texto, estilos predefinidos, referencias, tablas y posibilidad de enlazar datos con otros programas de esta suite hacen un que sea un procesador de textos muy completo y adecuado para crear y maquetar esta documentación siguiendo una plantilla existente.

Microsoft PowerPoint es un programa que esta suit ofrece para desarrollar, realizar y visualizar presentaciones con una variedad de diapositivas, ya sean creadas o disponibles, dinámicas o estáticas y la posibilidad de incluir dibujos, animaciones, videos y/o sonidos.

Se usará PowerPoint para la creación de las diapositivas usadas en la presentación.

Microsoft Project es el software que se ha usado para la administración del proyecto. Ofrece la posibilidad de

planificar el proyecto, crear tareas, asignar recursos, dar seguimiento del proyecto, administrar el presupuesto y analizar cargas de trabajo. Incluye gráficas Gantt, calendario y posibilidad de atribuir horarios a este asó como diagramas de red entre otras cosas.

Proyecto fue fundamental para el desarrollo del proyecto usando gran parte de sus características.

Ilustración 47 Logo Project

7.3.3.2 *Ventajas*

- Conocimiento del uso de la suite en muchos años.
- Calidad de trabajo.
- Gran variedad de funciones útiles.

7.3.3.3 *Inconvenientes*

- Uso de pago, aunque ofrece versión de prueba para estudiantes.

7.3.3.4 *Elección*

La elección de esta suite frente a otras se debe fundamentalmente a los años de experiencia que se tiene con respecto al uso de la misma y al gran resultado obtenido en estos años de uso.

La versión usada fue la 2007 en todos los programas debido a la instalación ya obtenida anteriormente en el PC que se realizó toda la documentación.

7.3.4 Enterprise Architect

Ilustración 48 Logo Enterprise Architect

7.3.4.1 *Descripción*

Es un software que proporciona las herramientas necesarias para la realización de diagramas siguiendo las especificaciones UML 2.3 desarrollado por Sparx Systems.

Con EA se puede modelar, gestionar, diseñar y visualizar los trece diagramas de UML así como alguno más extra. También incluye diagramas entidad-relación para representar bases de datos aunque en este caso no se usó.

Destacar la ingeniería inversa de código fuente que posee soportado para varios lenguajes entre los que se encuentra Java. Esta herramienta fue usada para la creación del diagrama de clases final ofrecido.

Se puede encontrar en su web documentación de cómo realizar cada diagrama y para qué sirve cada elemento con ejemplos visuales.

7.3.4.2 *Ventajas*

- Experiencia de trabajo con él.
- Buena documentación con ejemplos de cada diagrama y explicaciones útiles.
- Capacidad para crear todos los diagramas requeridos en esta documentación.

7.3.4.3 *Inconvenientes*

- Software de pago.

7.3.4.4 *Elección*

Una vez más la elección de este programa se debe principalmente a la experiencia de uso con el mismo, así como a la posibilidad de desarrollar todo los diagramas que esta documentación requería.

La versión utilizada para el desarrollo es la 7.0 debido, al igual que el paquete de Microsoft, a que era la versión instalada en el PC con el que se desarrolló la documentación del proyecto.

7.4 Creación del Sistema

7.4.1 Problemas Encontrados

7.4.1.1 *Tecnología Fi-Ware*

Al inicio del desarrollo del trabajo fin de máster, se realizó un estudio de la plataforma Fi-Ware con el fin de trabajar sobre ello y poder realizar una aplicación utilizando sus servicios.

Según las búsquedas realizadas, Fi-Ware es una plataforma fomentada por la Unión Europea y apoyada por algunas de las principales empresas TIC europeas como Telefónica, la cual que representa una opción abierta para el desarrollo y despliegue global de aplicaciones con Internet de las cosas.

Debido a que esta plataforma ofrece un conjunto de APIs abiertas y completamente libres de royalties para el desarrollo rápido de aplicaciones en numerosos sectores y que existe una gran inversión de dinero en el fomento de estas aplicaciones elegí trabajar sobre ella en mi proyecto.

http://www.fiware.org/

A lo largo del tiempo me fui dando cuenta de que esta plataforma que ofrecía todos estos servicios no daba una respuesta clara a los requisitos que mi proyecto necesitaba y después de muchas búsquedas me di cuenta de que esta plataforma aún estaba demasiado nueva y apenas se puede encontrar información de proyectos desarrollados sobre ella que ayuden a la compresión del uso de esta plataforma y sus servicios.

A continuación describiré algunos de los problemas encontrados que me llevaron al cambio de uso de tecnología.

- Para el desarrollo de las primeras pruebas con algunas de esas APIs o elementos básicos, que los desarrolladores de Fi-Ware llaman General Enablers, se podía hacer desde un laboratorio on-line llamado, Fi-Lab. http://www.fiware.org/lab/ . En este laboratorio on-line puedes encontrar un catálogo de los distintos General Enablers que hay, pero que a la hora de instanciarlos e instalarlos resultaba demasiado dificultoso, produciendo multitud de errores.
- Por cada cuenta de usuario se asignaba algunas direcciones IP públicas en las que podías instanciar distintos General Enablers desplegados en máquinas virtuales ya creados, pero habitualmente al acceder a estas máquinas se encontraban caídas sin ningún tipo de razón aparente y duraban en este estado largos periodos de tiempo, lo que provocaba muy poco seguridad en la ejecución de cualquier aplicación.
- Cada General Enabler estaba desarrollado en un lenguaje de programación diferente, lo que hacía que en muchas ocasiones no se podría instalar varios GE en una misma máquina e incluso la interconexión entre ellos no estaba bien definida.

Por estas razones y algunas otras decidí cambiar la orientación del desarrollo de mi proyecto hacia una tecnología que me ofreciera una mayor seguridad a l ahora de desarrollar cualquier prototipo o aplicación.

Respecto a esta plataforma creo que se trata de una plataforma demasiado nueva y que actualmente carece de información y de posibles soluciones son las que poder desarrollar.

Posiblemente en un futuro se puedan realizar el desarrollo de aplicaciones con esta plataforma y sin los problemas encontrados, pero en mi caso particular no podía esperar y gastar tanto tiempo en la investigación de la plataforma puesto que tenía una fecha límite de entrega que tenía que cumplirla.

7.4.1.2 *Lógica Difusa*

En la configuración de las variables lingüísticas tanto de entrada como de salida con las que trabaja la lógica difusa del prototipo se encontró el siguiente problema:

Los bloquees de lógica difusa que operan en el prototipo devuelven una salida que es un número concreto, este número concreto está dentro de unas variables lingüísticas definidas al inicio de los bloques de lógica difusa.

En el inicio del desarrollo se identificaron 4 variables lingüísticas para la salida de la sensación térmica, lo que indicaba que existía un gran rango de valores para cada variable lingüística, llegando a ocupar casi 20ºC de diferencia en un mismo rango, por lo que los valores de salida no eran muy exactos con respecto a sus variables lingüística.

Como medida de solución se crearon más variables lingüísticas en cada entrada y salida de los bloques de lógica difusa para así poder concretar más los valores finales y que cada variable lingüística englobe un rango menor de valores.

En la definición de las reglas difusas, para afinar lo máximo posible los resultados de las salidas de estas, se definieron todas las posibilidades, es decir todas las combinaciones posibles con respecto a las variables de entrada. Esto se realizó para los dos bloquees de reglas que se usaron.

7.4.1.3 *Conversor Analógico-Digital - MCP 3008*

En la instalación del sensor de temperatura TMP 36 usado en la Raspberry Pi ocurrió un erro a la hora de la lectura de datos.

Este sensor de temperatura ofrece sus datos de manera analógica, pero la Raspberry Pi no lee datos analógicos sino digitales y la única manera de que la Raspberry Pi pueda leer dichos datos ofrecidos por el sensor de temperatura utilizado serían usando un conversor.

La solución del problema fue instalar un conversor analógico-digital MCP3008 que nos ofrece la posibilidad de usar ocho pines de entradas analógicas donde podemos conectar nuestro pin de datos del sensor TMP a una de ellas para que la Raspberry Pi pueda trabajar con los datos obtenidos.

7.4.1.4 *Calibrar sensor de datos TMP36*

En la programación de la lectura de los datos del sensor de la Raspberry Pi se encontró un error a la hora de comprobarlos. Dichos datos se mostraban por consola con una alternancia de resultados bastante amplia llegando a veces a dar diferencias de ±10 grados centígrados, eso es debido a la poca calidad del sensor utilizado.

Para obtener una lectura de datos más regular y fiable se desarrolló un algoritmo de calibración para acercarnos a conseguir datos más realistas que fue el siguiente:

- Cada 100 milisegundos se realizó una media aritmética de las veces que el sensor de temperatura captaba el valor de temperatura.
- Cada media aritmética obtenida del punto anterior se agregó ordenadamente a un Array de 10 posiciones.
- Se eliminaron el primer y el último dato de esa lista, considerando que eran valores de mayor distancia con la media.
- Se calculó la media aritmética de cada Array con las 8 medias restantes y almacenadas. Considerando dicha media aritmética el valor adecuado a la hora de trabajar con la temperatura que el sensor de la Raspberry Pi ofrece.

7.4.2 Descripción Detallada de las Clases

Dentro de la carpeta *jFuzzy* del documento adjuntado a esta documentación se podrán ver una carpeta llamada *doc*, que contiene toda la información detallada de las clases generada con la herramienta correspondiente del entorno de desarrollo Eclipse, el Javadoc.

Capítulo 8. Desarrollo de las Pruebas

8.1 Pruebas Unitarias

Como se ha descrito anteriormente se han creado pruebas unitarias a través de *Junit*.

Las pruebas están desarrolladas en el paquete *com.test* del proyecto, y ahí podremos encontrar una clase *Tests.java* que lo que hace es cargar los dos archivos *fcl* que contienen toda la información de la lógica difusa.

Se han implementado dos métodos, uno para cada bloque de lógica difusa:

- Sensación térmica
- Salida Final

Respecto a las pruebas unitarias, se han creado doce test, cinco para la sensación térmica y siete para la salida final.

Cada test de lógica difusa contiene cinco métodos que comprueban el resultado de los valores mostrando todas las posibilidades de combinaciones de variables lingüísticas de entrada que son posibles, en el caso de la salida final cada test contiene siete combinaciones posibles.

De esta manera se ha realizado tantas pruebas unitarias como reglas por cada paquete de lógica difusa tienen, es decir:

- 35 combinaciones para la sensación térmica.
- 49 combinaciones para la salida final.

Los datos introducidos han sido datos aleatorios pero siempre comprobando que eran datos que se encontraban dentro del rango de la variable lingüística asociada.

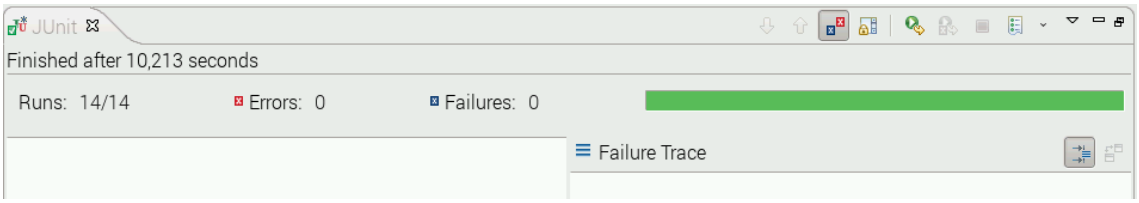


Ilustración 49 Resumen pruebas unitarias

Capítulo 9. Manuales del Sistema

9.1 Manual de Instalación

Elaborar un manual que contemple todos los pasos necesarios para instalar nuestro sistema, incluyendo la instalación de otras herramientas o software cualquiera (sea o no comercial) necesario para que funcione. Debemos explicarlo todo paso a paso de forma clara y acompañarlo por capturas de pantalla adecuadas.

9.1.1 Instalación de Raspberry

Para la instalación de la Raspberry se añadirá a continuación información y los pasos que se siguieron.

En primer lugar se compró una Raspberry con algunos de los elementos necesarios para su funcionamiento y el de los sensores correspondientes.

- Raspberry Pi 2 Model B.
- Cable de corriente
- Adaptador USB Wi-fi
- Cable adaptador 1080 HDMI a VGA
- Kit de componentes para trabajar con sensores.
    - Breadboard
    - Resistencias
    - Led's's
    - RGB Led's's
    - Termómetro TMP 36
    - Capacitador
    - Fotoresistor
    - Cables Jumper para las conexiones
    - Switch
- Tarjeta micro SD.

Lo primero que debemos de tener en cuenta es que se debe conectar a la Raspberry Pi un teclado, un ratón, y un monitor que podremos conectarlo a través su salida HDMI o a través del adaptador HDMI a VGA que antes se ha mencionado, además del adaptador Wi-fi para la conexión a Internet o simplemente conectando un cable Ethernet RJ45 típico y su cable de corriente correspondiente.

9.1.1.1 *Instalación Software*

Para la instalación del sistema operativo en este caso, se siguieron las indicaciones referidas en https://www.raspberrypi.org desde donde se descargó el sistema operativo *Raspbian,* que fue instalado en la tarjeta micro SD para usarse en la Raspberry Pi.

Una vez arrancada, antes de usar el comando *startx* para la activación de la interfaz gráfica, podemos configurar las características con:

sudo raspi-config.

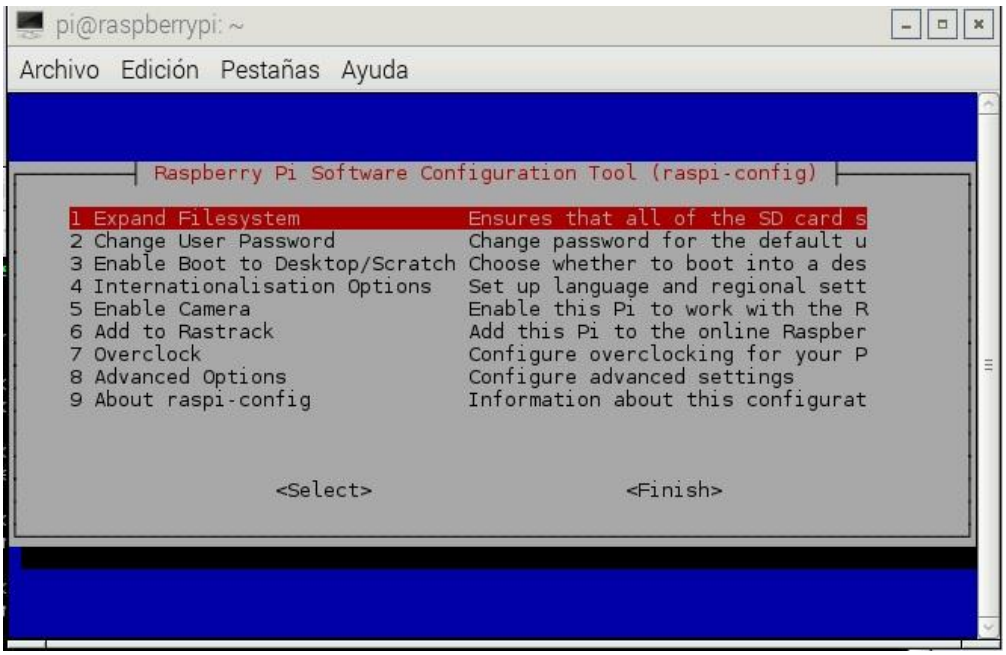


Ilustración 50 Panel de configuración de Raspberry Pi.

http://raspberryparatorpes.net/empezando/raspi-config-configuracion-inicial-de-raspbian

Desde esta pantalla podremos realizar una configuración inicial de la Raspberry llegando a características como la configuración del teclado o cambiar la contraseña del microordenador.

En el caso del desarrollo de este prototipo se instaló un entorno de desarrollo para programar con el lenguaje de programación Java, el entorno fue Eclipse. A continuación se muestran algunos comandos para la instalación del IDE de Eclipse.

http://www.rpiblog.com/2014/03/installing-oracle-jdk-8-on-raspberry-pi.html

- Actualización del sistema:

  $ sudo apt-get update

  $ sudo apt-get upgrade

- *Instalación de Eclipse*

  $ Sudo apt-get install eclipse

- *Descarga del JDK 8 para ARM.*

  http://www.oracle.com/technetwork/java/javase/downloads/jdk8-arm-downloads-2187472.html

- Descomprime el archive descargado a un directorio.

  $ sudo tar zxvf jdk-8-linux-arm-vfp-hflt.tar.gz -C /directory

- Establece por defecto el Nuevo JDK8 Java y Javac que la raspberry usará

  $ sudo update-alternatives --install /usr/bin/javac javac /opt/jdk1.8.0/bin/javac 1
  $ sudo update-alternatives --install /usr/bin/java java /opt/jdk1.8.0/bin/java 1

  $ sudo update-alternatives --config javac
  $ sudo update-alternatives --config java

- Verifica la versión que se está usando.

  $ java -version
  $ javac –version

Para poder trabajar desde eclipse con los sensores y actuadores instalados en la placa Beardboard se configurará el proyecto del eclipse correspondiente con la librerías pi4j, la cual permiten acceder a los pines de comunicación GPIO de la Raspberry.

Las podremos descargar desde http://pi4j.com/download.html

9.1.1.2 *Instalación Hardware*

9.1.1.2.1 Sensor de temperatura TMP 36 y conversor analógico digital MCP3008.

- En primer lugar se muestra la disposición de los pines de la Raspberry PI que se utiliza.

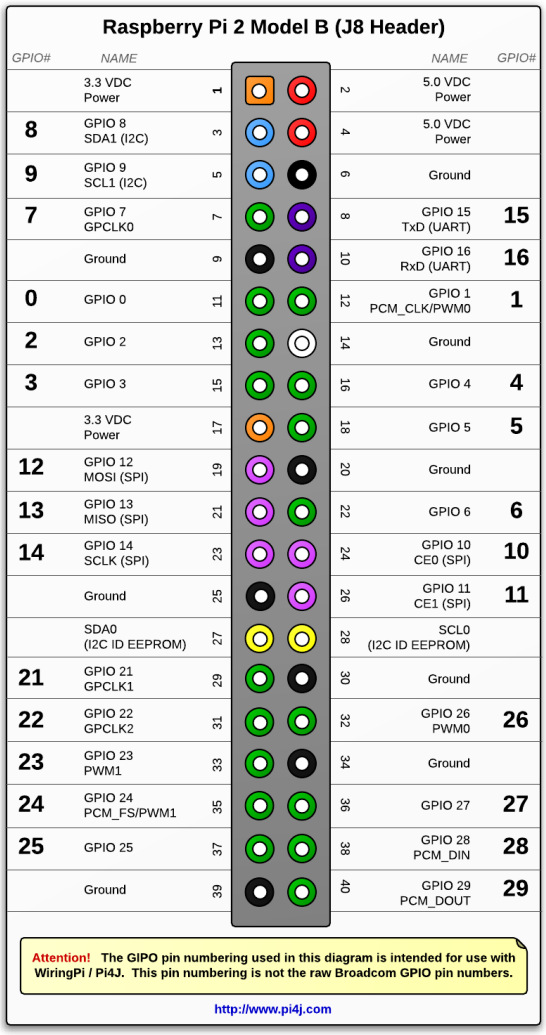


Ilustración 51 Esquema Pin GPIO Raspberry Pi 2 Model B

En este caso, se necesita de un conversor analógico digital cuya conexión a la es la siguiente.

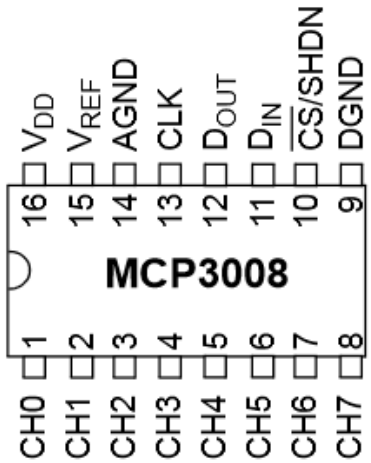


Ilustración 52 Esquema MCP3008

Esta imagen muestra que de los pines 9 al 16 tienen una configuración concreta y que por otro lado de los pines 1 al 8 son los ocho canales de entrada que dispone el conversor.

En nuestro caso las conexiones fueron establecidas de la siguiente manera:

Conexión MCP3008	PIN GPIO
VDD	3.3 V
VREF	3.3 V
AGND	Ground
CLK	GPIO 1
DOUT	GPIO 4
DIN	GPIO 5
CS/SHDN	GPIO 6
DGND	Ground

Tabla 38 Conexiones MCP 3008 con Raspberry Pi 2 Model B

El sensor de temperatura TMP36 tiene la siguiente predisposición de pines:

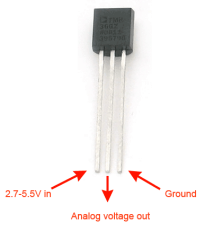


Ilustración 53 Sensor de temperatura TMP36

Como podemos observar la tabla de conexiones sería así.

TMP36	Conexión

2,7-5,5 V in	3.3 V
Analog voltaje out	CH0 (MCP3008)
Ground	Ground

Tabla 39 Conexiones sensor TMP36 con Raspberry Pi 2 Model B

A continuación mostraré una imagen de cómo quedó instalado la placa con sus correspondientes sensores y conexiones en la Raspberry Pi.

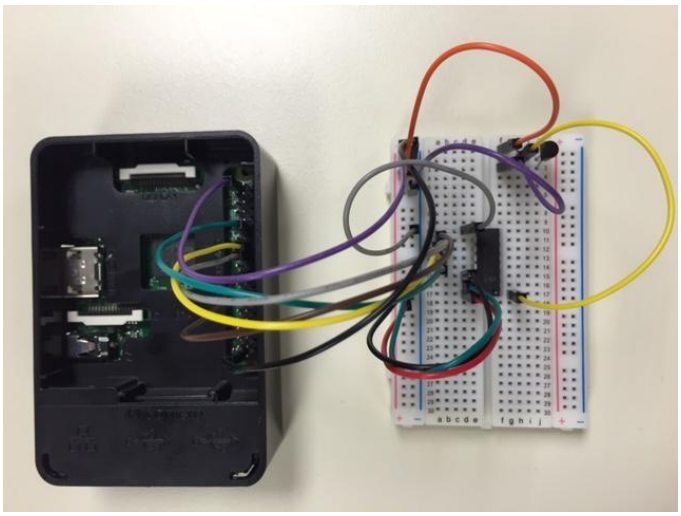


Ilustración 54 Ejemplo conexión MCP3008, Sensor TMP36 y Raspberry PI 2 Model B

9.1.1.2.2 Led's's actuadores.

Se instalaron 5 Led's's. Dos de ellos rojos, dos de ellos amarillo y un RGB.

En nuestro caso los 4 Led´s simples se instalarán de la misma manera, es decir la pata positiva irá conectada a un PIN GPIO de la Raspberry pasando antes por una resistencia de 470Ω y la pata negativa a la toma de tierra de la Raspberry Pi.

En el caso del LED tricolor RGB, está formado por 4 patas, la más larga de ellas estará conectada a la toma de tierra y las otras tres, a tres PIN GPIO respectivamente uno por cada color, rojo, verde y azul.

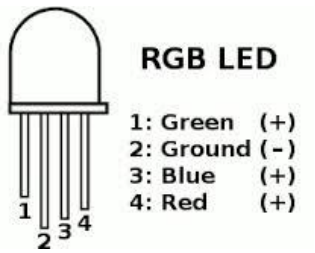


Ilustración 55 Esquema LED RGB

En la siguiente tabla se explica la conexión de los LED con los PIN GPIO de la Raspberry.

LED	PIN GPIO
LED 1 ROJO	GPIO 3
LED 2 ROJO	GPIO 0
LED 3 AMARILLO	GPIO 25
LED 4 AMARILLO	GPIO 2
RGB ROJO	GPIO 24
RGB VERDE	GPIO 23
RGB AZUL	GPIO 22

Tabla 40 Conexiones LED RGB con Raspberry PI 2 Model B

La nomenclatura de los Led's's se ha tomado de izquierda a derecha es decir el LED1 Rojo es el LED rojo de mas a la izquierda y así sucesivamente.

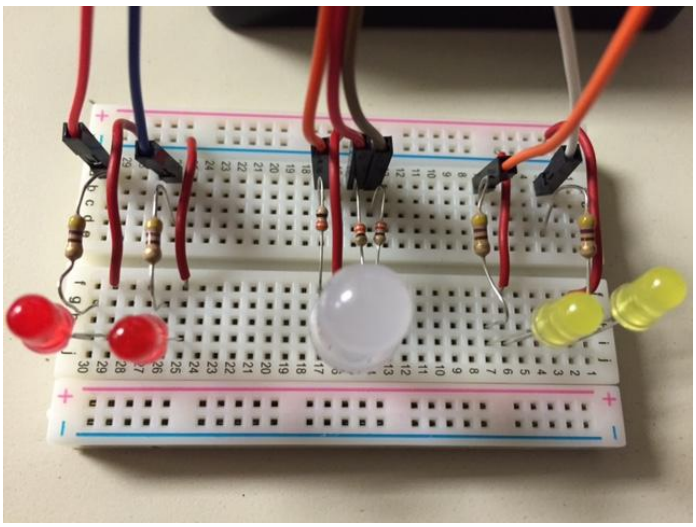


Ilustración 56 Ejemplo de conexión Led's y Raspberry Pi 2 Model B

La instalación y configuración final del prototipo se puede ver en la siguiente imagen.

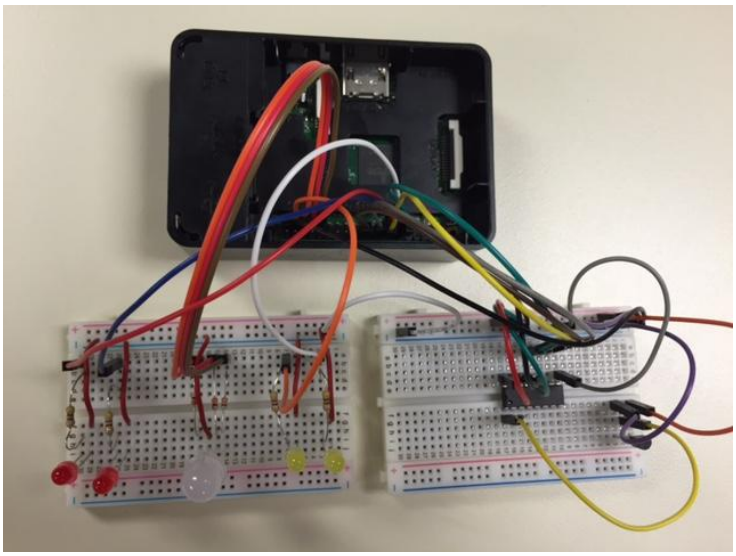


Ilustración 57 Ejemplo completo de la instalación del prototipo

9.2 Manual de Ejecución

En el caso de este prototipo el manual de ejecución es algo muy sencillo, ya que por ahora se limita al arranque de la aplicación desde el Eclipse instalado en la Raspberry.

La clase principal que pone en funcionamiento el prototipo se llama jFuzzyClass.java.

Una vez arrancado la aplicación nos aparecerá un menú por consola donde elegiremos la opción correspondiente.

```
Elige opción:
1.- Ejecutar Valores Reales ininterrumpidamente
2.- Ejecutar valores Unitarios
3.- Ejecutar Arrays de tamaño introducido por usuario
4.- Ejecutar Arrays semialeatorios
5.- Ejecutar hilo con valores aleatorios
0.- Salir
```

Ilustración 58 Menú del usuario al arrancar el prototipo

9.3 Manual de Usuario

En esta sección veremos las funcionalidades que tiene el prototipo y que ocurre en cada una de ellas para que se pueda comprobar el funcionamiento del mismo

Una vez iniciado el prototipo nos aparecerá un menú donde podemos ver las funcionalidades el prototipo como se ha explicado en el Manual de ejecución.

A continuación se identificará el funcionamiento de los ítems del menú.

9.3.1 Ejecutar valores reales ininterrumpidamente

En esta sección del menú, el prototipo se pondrá en marcha sin pausa y consumiendo datos reales de las distintas fuentes establecidas.

Para el cálculo de la sensación térmica se obtendrán los siguientes datos:

- Temperatura exterior desde red de internet de las cosas ThingSpeak.
- Humedad exterior desde red de internet de las cosas Midgar.

Para el cálculo de la salida final que activará los actuadores según convenga.

- Sensación térmica desde salida de lógica difusa de la misma aplicación.
- Temperatura local desde sensor instalado en Raspberry Pi.

El prototipo mostrará en la salida los datos obtenidos y los resultados de los cálculos con lógica difusa tantas veces como se deje ejecutar.

```
**************************************

Temperatura interior obtenida por sensor Raspberry: 18,32
Temperatura Exterior obtenida de MIDGAR :15,44
Humedad Exterior obtenida de ThnigSpeak :27,64

SENSACION TERMICA: 18,75

VALOR DE SALIDA FINAL: 20,00
**************************************
```

Ilustración 59 Valores de menú 1

Esta salida se mostrará con los controladores asignados, en el ejemplo anterior se muestra a continuación:

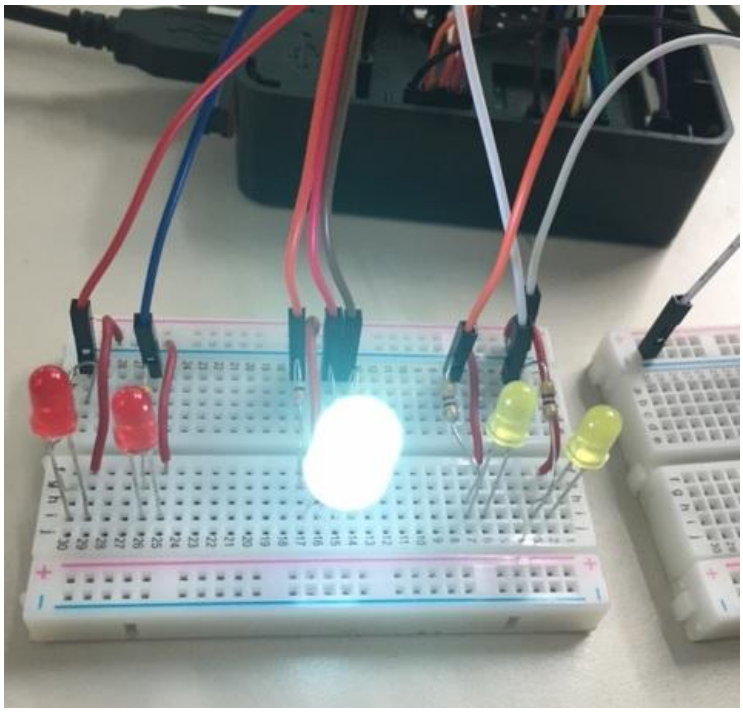


Ilustración 60 Salida Led's Menú 1

9.3.2 Ejecutar valores unitarios

Esta sección del menú pondrá en marcha el prototipo con los datos introducidos por el usuario que serán pedidos por pantalla.

Esta sección solamente ejecutará el prototipo una vez, y mostrará una información mucho más detallada del proceso como se verá a continuación:

Después de introducir los datos, y de calcular la sensación térmica aproximada correspondiente, vemos que se muestran los valores de pertenencia de la salida de la sensación térmica, es decir, el grado de verdad de la variable de salida lingüística al que pertenece. En el ejemplo se muestra la sensación es Extremadamente alta en un grado de verdad de 0.715 dentro de la variable lingüística que viene detallada a continuación, como un triángulo de valores que comienza en 35 grados y finaliza en 50 grados cuyo vértice que se corresponde con el valor de verdad absoluto sería 47.

También se muestra la variable *Defuzzier* que corresponde al método de *defucificación* utilizan que en este caso es el de centro de gravedad, usándose así en todos los caso del prototipo ya que es la función más usada y que representa mejor el valor que queremos representar.

El *Default Value* que se ve, corresponde al valor por defecto que se mostrará si existe cualquier error en el cálculo o si la variable de salida no está definida en ninguna variable lingüística asociada.

```
Introduce temperatura exterior:
34.5
Introduce humedad exterior :
64.5
Introduce temperatura interior :
29.4



SENSACION TERMICA: 43,58
Valores de pertenencia :  ext_cond :
        Defuzzifier : CenterOfGravity
        Latest defuzzified value: 43.5808673700418
        Default value: 0.0
        Term: normal    0.0     Triangular : 14.0 , 18.0 , 24.0
        Term: high      0.0     Triangular : 22.0 , 27.0 , 30.0
        Term: extremHigh        0.7150722808368167      Triangular : 35.0 , 47.0 , 50.0
        Term: extrmLow  0.0     Triangular : -15.0 , -10.0 , -5.0
        Term: low       0.0     Triangular : 0.0 , 5.0 , 18.0
        Term: veryLow   0.0     Triangular : -7.0 , -2.0 , 3.0
        Term: veryHigh  0.0     Triangular : 28.0 , 33.0 , 38.0
```

Ilustración 61 Menú 2-1

La siguiente información que se muestra es la lista completa de todas las reglas difusas por las que estas dos variables pasan y su grado de verdad en cada una de ellas.

Se puede comprobar que la regla número 28 y 29 son las únicas reglas cuyos valores dados están dentro de las variables lingüísticas establecidas.

Estas reglas serán las que se activan con el grado de verdad correspondiente para el cálculo de la salida de la sensación térmica.

```
1       (0.0)   if (temp_arduino IS extrmLow) AND (hum_thinkSpeak IS veryLow) then ext_cond IS extrmLow [weight: 1.
2       (0.0)   if (temp_arduino IS extrmLow) AND (hum_thinkSpeak IS low) then ext_cond IS extrmLow [weight: 1.0]
3       (0.0)   if (temp_arduino IS extrmLow) AND (hum_thinkSpeak IS normal) then ext_cond IS extrmLow [weight: 1.0
4       (0.0)   if (temp_arduino IS extrmLow) AND (hum_thinkSpeak IS high) then ext_cond IS veryLow [weight: 1.0]
5       (0.0)   if (temp_arduino IS extrmLow) AND (hum_thinkSpeak IS veryHigh) then ext_cond IS low [weight: 1.0]
6       (0.0)   if (temp_arduino IS veryLow) AND (hum_thinkSpeak IS veryLow) then ext_cond IS veryLow [weight: 1.0]
7       (0.0)   if (temp_arduino IS veryLow) AND (hum_thinkSpeak IS low) then ext_cond IS veryLow [weight: 1.0]
8       (0.0)   if (temp_arduino IS veryLow) AND (hum_thinkSpeak IS normal) then ext_cond IS veryLow [weight: 1.0]
9       (0.0)   if (temp_arduino IS veryLow) AND (hum_thinkSpeak IS high) then ext_cond IS low [weight: 1.0]
10      (0.0)   if (temp_arduino IS veryLow) AND (hum_thinkSpeak IS veryHigh) then ext_cond IS low [weight: 1.0]
11      (0.0)   if (temp_arduino IS low) AND (hum_thinkSpeak IS veryLow) then ext_cond IS veryLow [weight: 1.0]
12      (0.0)   if (temp_arduino IS low) AND (hum_thinkSpeak IS low) then ext_cond IS low [weight: 1.0]
13      (0.0)   if (temp_arduino IS low) AND (hum_thinkSpeak IS normal) then ext_cond IS low [weight: 1.0]
14      (0.0)   if (temp_arduino IS low) AND (hum_thinkSpeak IS high) then ext_cond IS low [weight: 1.0]
15      (0.0)   if (temp_arduino IS low) AND (hum_thinkSpeak IS veryHigh) then ext_cond IS normal [weight: 1.0]
16      (0.0)   if (temp_arduino IS normal) AND (hum_thinkSpeak IS veryLow) then ext_cond IS normal [weight: 1.0]
17      (0.0)   if (temp_arduino IS normal) AND (hum_thinkSpeak IS low) then ext_cond IS normal [weight: 1.0]
18      (0.0)   if (temp_arduino IS normal) AND (hum_thinkSpeak IS normal) then ext_cond IS normal [weight: 1.0]
19      (0.0)   if (temp_arduino IS normal) AND (hum_thinkSpeak IS high) then ext_cond IS normal [weight: 1.0]
20      (0.0)   if (temp_arduino IS normal) AND (hum_thinkSpeak IS veryHigh) then ext_cond IS normal [weight: 1.0]
21      (0.0)   if (temp_arduino IS high) AND (hum_thinkSpeak IS veryLow) then ext_cond IS normal [weight: 1.0]
22      (0.0)   if (temp_arduino IS high) AND (hum_thinkSpeak IS low) then ext_cond IS high [weight: 1.0]
23      (0.0)   if (temp_arduino IS high) AND (hum_thinkSpeak IS normal) then ext_cond IS veryHigh [weight: 1.0]
24      (0.0)   if (temp_arduino IS high) AND (hum_thinkSpeak IS high) then ext_cond IS veryHigh [weight: 1.0]
25      (0.0)   if (temp_arduino IS high) AND (hum_thinkSpeak IS veryHigh) then ext_cond IS extremHigh [weight: 1.0
26      (0.0)   if (temp_arduino IS veryHigh) AND (hum_thinkSpeak IS veryLow) then ext_cond IS high [weight: 1.0]
27      (0.0)   if (temp_arduino IS veryHigh) AND (hum_thinkSpeak IS low) then ext_cond IS veryHigh [weight: 1.0]
28      (0.55)  if (temp_arduino IS veryHigh) AND (hum_thinkSpeak IS normal) then ext_cond IS extremHigh [weight: 1
29      (0.45)  if (temp_arduino IS veryHigh) AND (hum_thinkSpeak IS high) then ext_cond IS extremHigh [weight: 1.0
30      (0.0)   if (temp_arduino IS veryHigh) AND (hum_thinkSpeak IS veryHigh) then ext_cond IS extremHigh [weight:
31      (0.0)   if (temp_arduino IS extremHigh) AND (hum_thinkSpeak IS veryLow) then ext_cond IS veryHigh [weight:
32      (0.0)   if (temp_arduino IS extremHigh) AND (hum_thinkSpeak IS low) then ext_cond IS extremHigh [weight: 1.
33      (0.0)   if (temp_arduino IS extremHigh) AND (hum_thinkSpeak IS normal) then ext_cond IS extremHigh [weight:
```

Ilustración 62 Menú 2-2

Para una mejor representación de los datos esta sección del menú también muestra una disposición gráfica de las variables lingüísticas establecidas, tanto las de salida como las de entrada, y su correspondiente salida representada en otro gráfico como se ve a continuación,

- Variable lingüística de entrada, temperatura exterior recogida por el Arduino.

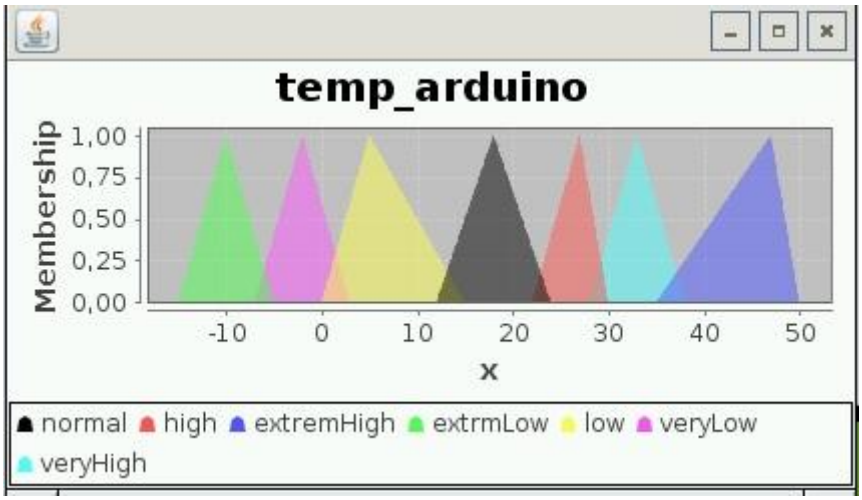


Ilustración 63 Gráfico temperatura exterior

- Variable lingüística de entrada, humedad exterior recogida desde ThingSpeak.

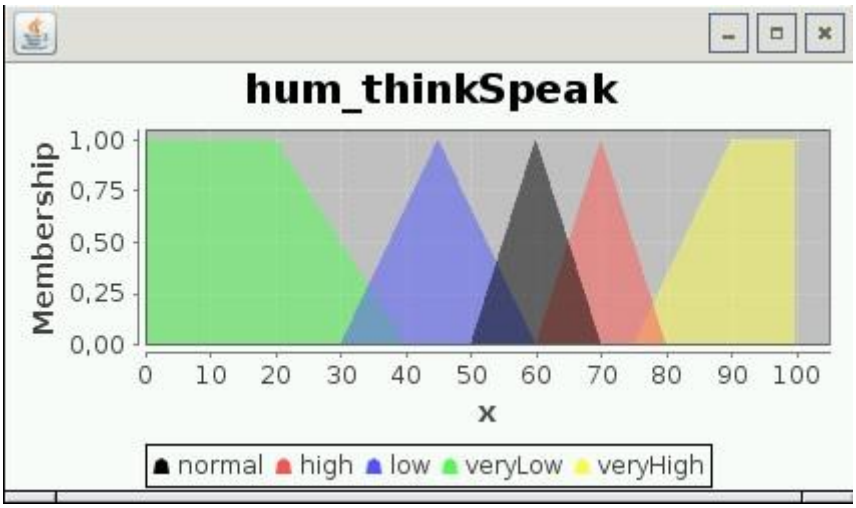


Ilustración 64 Gráfico humedad exterior.

- Variable lingüística de salida, condiciones exteriores es decir sensación térmica.

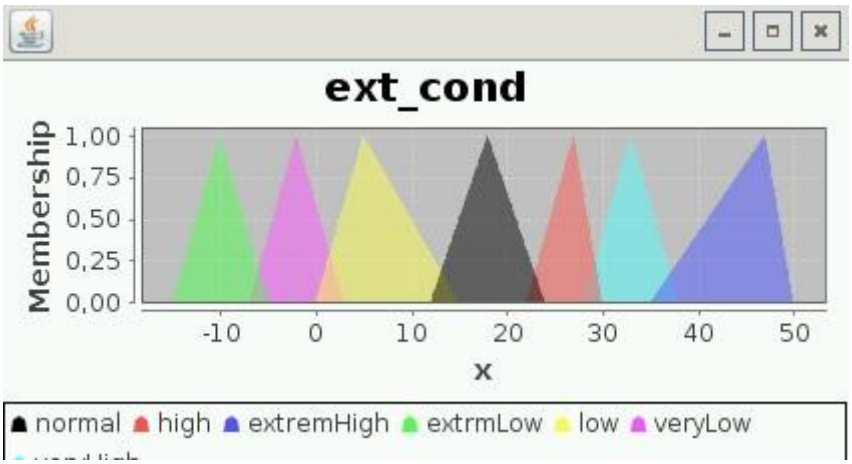


Ilustración 65 Gráfico sensación térmica

- Valor de sensación térmica calculado a través de la lógica difusa.

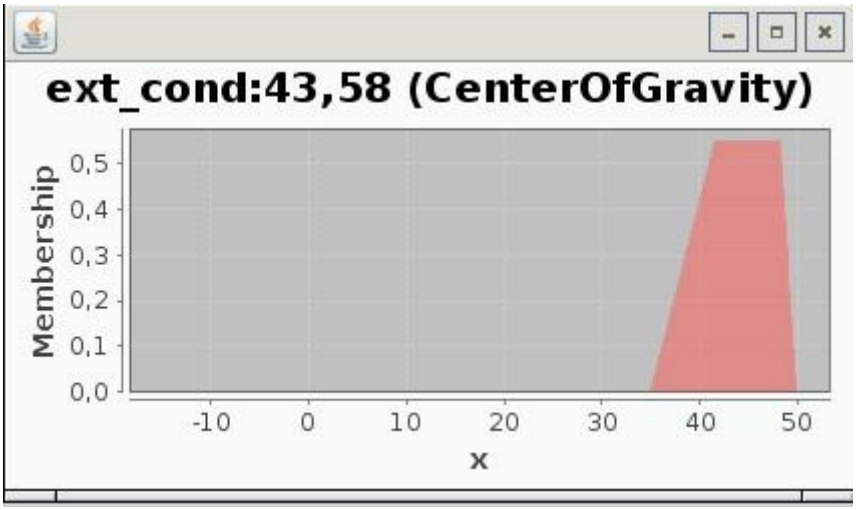


Ilustración 66 Gráfico con el valor de Sensación térmica.

Por otro lado se realizará el proceso cálculo que conlleva la obtención de la salida final que irá a los controladores, a continuación se mostrará las imágenes del proceso, que tendrá la misma dinámica de ejecución excepto que una variable de entrada será la salida de la sensación térmica obtenida y la otra la temperatura local.

- Valores de pertenencia

```
VALOR DE SALIDA FINAL: 38,69
Valores de pertenencia :  signal :
        Defuzzifier : CenterOfGravity
        Latest defuzzified value: 38.68736843780217
        Default value: 0.0
        Term: on2       0.0     PieceWiseLinear : (-15.0, 1.0) , (10.0, 1.0) , (15.0, 0.0) ;
        Term: stable    0.0     Triangular : 18.0 , 20.0 , 22.0
        Term: off2      1.0     PieceWiseLinear : (25.0, 0.0) , (30.0, 1.0) , (50.0, 1.0) ;
        Term: off       0.0     Triangular : 20.0 , 25.0 , 30.0
        Term: on        0.0     Triangular : 10.0 , 15.0 , 20.0
```

Ilustración 67 Menú 2-3

- Reglas difusas

```
1       (0.0)   if (ext_cond IS extrmLow) AND (temp_local IS extrmLow) then signal IS on2 [weight: 1.0]
2       (0.0)   if (ext_cond IS extrmLow) AND (temp_local IS veryLow) then signal IS on2 [weight: 1.0]
3       (0.0)   if (ext_cond IS extrmLow) AND (temp_local IS low) then signal IS on2 [weight: 1.0]
4       (0.0)   if (ext_cond IS extrmLow) AND (temp_local IS normal) then signal IS on [weight: 1.0]
5       (0.0)   if (ext_cond IS extrmLow) AND (temp_local IS high) then signal IS stable [weight: 1.0]
6       (0.0)   if (ext_cond IS extrmLow) AND (temp_local IS veryHigh) then signal IS off [weight: 1.0]
7       (0.0)   if (ext_cond IS extrmLow) AND (temp_local IS extremHigh) then signal IS off2 [weight: 1.0]
8       (0.0)   if (ext_cond IS veryLow) AND (temp_local IS extrmLow) then signal IS on2 [weight: 1.0]
9       (0.0)   if (ext_cond IS veryLow) AND (temp_local IS veryLow) then signal IS on2 [weight: 1.0]
10      (0.0)   if (ext_cond IS veryLow) AND (temp_local IS low) then signal IS on2 [weight: 1.0]
11      (0.0)   if (ext_cond IS veryLow) AND (temp_local IS normal) then signal IS on [weight: 1.0]
12      (0.0)   if (ext_cond IS veryLow) AND (temp_local IS high) then signal IS stable [weight: 1.0]
13      (0.0)   if (ext_cond IS veryLow) AND (temp_local IS veryHigh) then signal IS off [weight: 1.0]
14      (0.0)   if (ext_cond IS veryLow) AND (temp_local IS extremHigh) then signal IS off2 [weight: 1.0]
15      (0.0)   if (ext_cond IS low) AND (temp_local IS extrmLow) then signal IS on2 [weight: 1.0]
16      (0.0)   if (ext_cond IS low) AND (temp_local IS veryLow) then signal IS on2 [weight: 1.0]
17      (0.0)   if (ext_cond IS low) AND (temp_local IS low) then signal IS on2 [weight: 1.0]
18      (0.0)   if (ext_cond IS low) AND (temp_local IS normal) then signal IS on [weight: 1.0]
19      (0.0)   if (ext_cond IS low) AND (temp_local IS high) then signal IS stable [weight: 1.0]
20      (0.0)   if (ext_cond IS low) AND (temp_local IS veryHigh) then signal IS off [weight: 1.0]
21      (0.0)   if (ext_cond IS low) AND (temp_local IS extremHigh) then signal IS off2 [weight: 1.0]
22      (0.0)   if (ext_cond IS normal) AND (temp_local IS extrmLow) then signal IS on2 [weight: 1.0]
23      (0.0)   if (ext_cond IS normal) AND (temp_local IS veryLow) then signal IS on2 [weight: 1.0]
24      (0.0)   if (ext_cond IS normal) AND (temp_local IS low) then signal IS on [weight: 1.0]
25      (0.0)   if (ext_cond IS normal) AND (temp_local IS normal) then signal IS stable [weight: 1.0]
26      (0.0)   if (ext_cond IS normal) AND (temp_local IS high) then signal IS stable [weight: 1.0]
27      (0.0)   if (ext_cond IS normal) AND (temp_local IS veryHigh) then signal IS off [weight: 1.0]
28      (0.0)   if (ext_cond IS normal) AND (temp_local IS extremHigh) then signal IS off2 [weight: 1.0]
29      (0.0)   if (ext_cond IS high) AND (temp_local IS extrmLow) then signal IS on2 [weight: 1.0]
30      (0.0)   if (ext_cond IS high) AND (temp_local IS veryLow) then signal IS on2 [weight: 1.0]
31      (0.0)   if (ext_cond IS high) AND (temp_local IS low) then signal IS on [weight: 1.0]
32      (0.0)   if (ext_cond IS high) AND (temp_local IS normal) then signal IS stable [weight: 1.0]
33      (0.0)   if (ext_cond IS high) AND (temp_local IS high) then signal IS off [weight: 1.0]
34      (0.0)   if (ext_cond IS high) AND (temp_local IS veryHigh) then signal IS off2 [weight: 1.0]
```

Ilustración 68 Menú 2-4

- Reglas difusas 47 y 48 activadas.

```
35      (0.0)   if (ext_cond IS high) AND (temp_local IS extremHigh) then signal IS off2 [weight: 1.0]
36      (0.0)   if (ext_cond IS veryHigh) AND (temp_local IS extrmLow) then signal IS on2 [weight: 1.0]
37      (0.0)   if (ext_cond IS veryHigh) AND (temp_local IS veryLow) then signal IS on2 [weight: 1.0]
38      (0.0)   if (ext_cond IS veryHigh) AND (temp_local IS low) then signal IS on [weight: 1.0]
39      (0.0)   if (ext_cond IS veryHigh) AND (temp_local IS normal) then signal IS stable [weight: 1.0]
40      (0.0)   if (ext_cond IS veryHigh) AND (temp_local IS high) then signal IS off [weight: 1.0]
41      (0.0)   if (ext_cond IS veryHigh) AND (temp_local IS veryHigh) then signal IS off2 [weight: 1.0]
42      (0.0)   if (ext_cond IS veryHigh) AND (temp_local IS extremHigh) then signal IS off2 [weight: 1.0]
43      (0.0)   if (ext_cond IS veryHigh) AND (temp_local IS extrmLow) then signal IS on2 [weight: 1.0]
44      (0.0)   if (ext_cond IS veryHigh) AND (temp_local IS veryLow) then signal IS on2 [weight: 1.0]
45      (0.0)   if (ext_cond IS veryHigh) AND (temp_local IS low) then signal IS on [weight: 1.0]
46      (0.0)   if (ext_cond IS veryHigh) AND (temp_local IS normal) then signal IS stable [weight: 1.0]
47      (0.0)   if (ext_cond IS veryHigh) AND (temp_local IS high) then signal IS off [weight: 1.0]
48      (0.0)   if (ext_cond IS veryHigh) AND (temp_local IS veryHigh) then signal IS off2 [weight: 1.0]
49      (0.0)   if (ext_cond IS veryHigh) AND (temp_local IS extremHigh) then signal IS off2 [weight: 1.0]
43      (0.0)   if (ext_cond IS extremHigh) AND (temp_local IS extrmLow) then signal IS on2 [weight: 1.0]
44      (0.0)   if (ext_cond IS extremHigh) AND (temp_local IS veryLow) then signal IS on2 [weight: 1.0]
45      (0.0)   if (ext_cond IS extremHigh) AND (temp_local IS low) then signal IS on [weight: 1.0]
46      (0.0)   if (ext_cond IS extremHigh) AND (temp_local IS normal) then signal IS off [weight: 1.0]
47      (0.14301445616736372)   if (ext_cond IS extremHigh) AND (temp_local IS high) then signal IS off2 [weigh
48      (0.20022023863430846)   if (ext_cond IS extremHigh) AND (temp_local IS veryHigh) then signal IS off2 [w
49      (0.0)   if (ext_cond IS extremHigh) AND (temp_local IS extremHigh) then signal IS off2 [weight: 1.0]
```

Ilustración 69 Menú 2-5

- Variable lingüística de entrada, sensación térmica, calculada anteriormente.

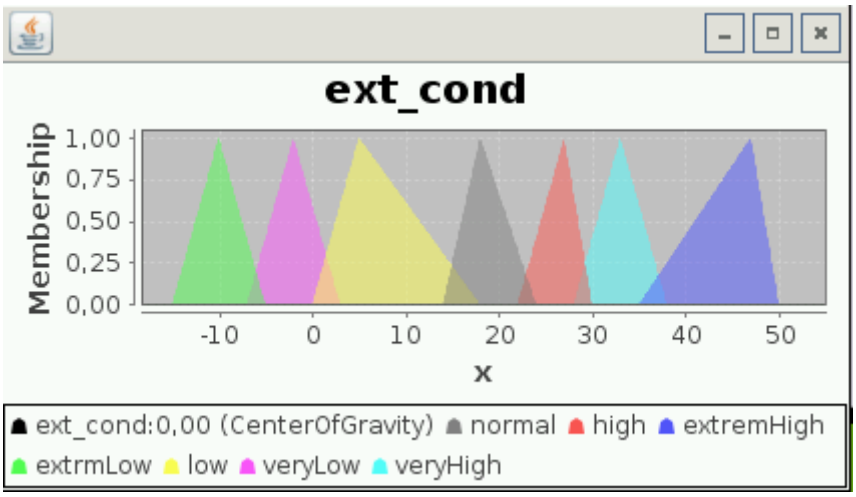


Ilustración 70 Gráfico sensación térmica

- Variable lingüística de entrada, temperatura local recogida por la Raspberry.

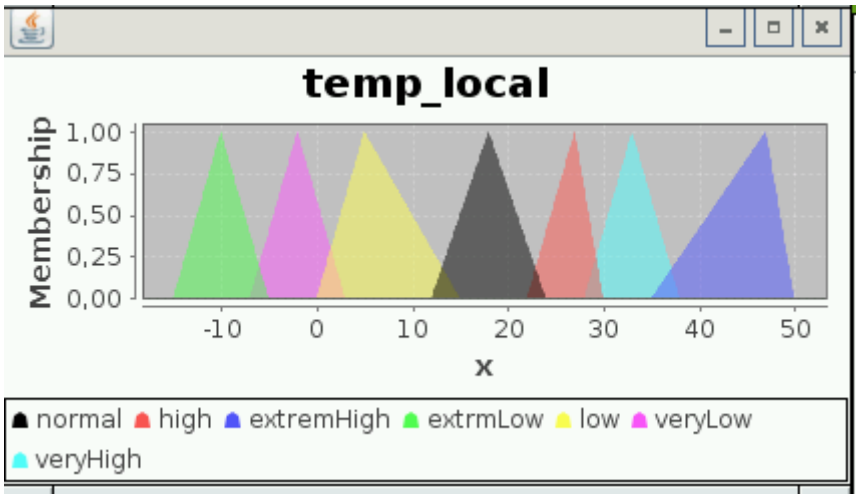


Ilustración 71 Gráfico temperatura local

- Variable lingüística de salida, salida final.

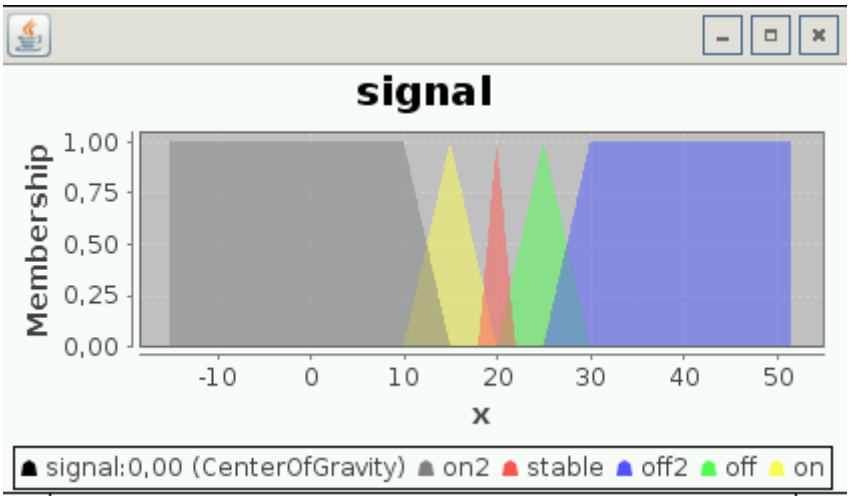


Ilustración 72 Gráfico de salida final

- Valor de la salida final calculado a través de la lógica difusa.

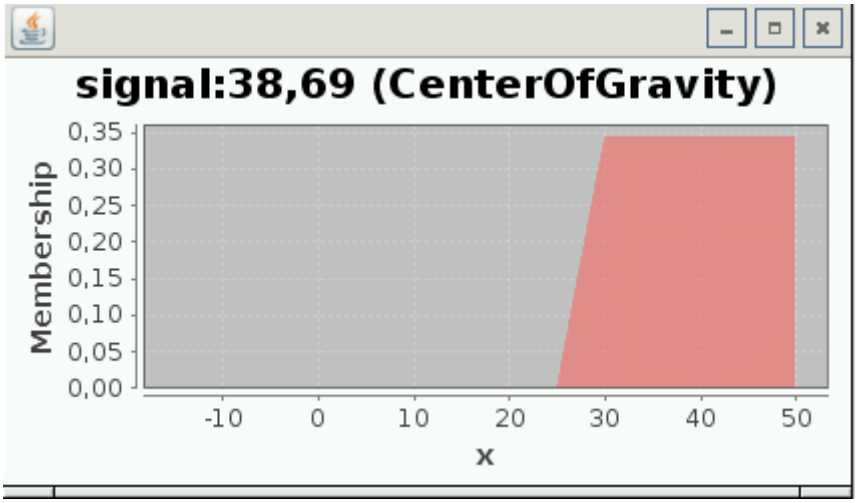


Ilustración 73 Valor de salida final

Una vez conseguido el valor final, se tratará para activar los controladores correspondientes, que en este caso al considerar una sensación térmica elevada y una temperatura local también elevada, nuestro prototipo acciona los dos Led's amarillo que se corresponderían con una doble potencia de un hipotético aire acondicionado simulado.

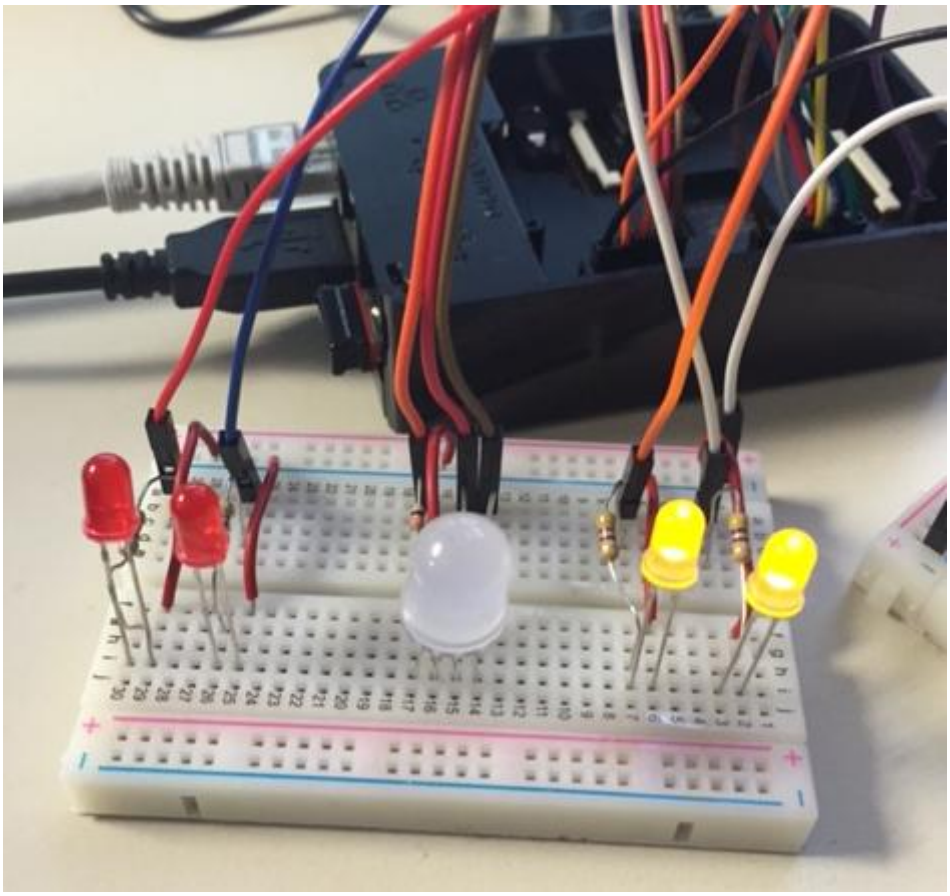


Ilustración 74 Activación de controladores

9.3.3 Ejecutar Arrays de tamaño introducido por el usuario.

En esta sección se mostrará el comportamiento del prototipo a través de ejecutar valores aleatorios creados en un Array de datos cuyo tamaño se pedirá por pantalla al usuario.

El usuario introducirá un número entero aleatorio y se crearán tres Arrays de ese tamaño con datos aleatorios uno para cada variable de entrada:

- Temperatura exterior.
- Humedad exterior.
- Temperatura local.

Los datos introducidos serán datos aleatorios sin ningún tipo de relación entre ellos, simplemente estarán acotados con dos rangos.

- Temperatura exterior e interior: -15° a 50°
- Humedad exterior: 0% a 100 %

El prototipo se ejecutará tantas veces como el tamaño de los Arrays hayan sido establecidos, cada vez usará el dato de una posición de cada uno de los tres Arrays creados.

Por último se pasarán un Array con los datos de cada salida final al controlador de Led´s de la Raspberry Pi y esta los mostrará secuencialmente.

En la consola podremos ver los datos de los Arrays creados así como los datos calculados de las sensaciones térmicas y las salidas finales.

```
Introduce tamaño de los Arrays que quieras crear:
2


LISTA TEMPERATURA EXTERIOR INICIAL

[21.65, 36.7]

LISTA HUMEDAD EXTERIOR INICIAL

[45.09, 70.71]


LISTA TEMPERATURA LOCAL INICIAL

[22.22, 7.57]

Valor de sensación térmica. Posición 0 del Array: 18,82
Valor de salida final. Posición 0 del Array: 20,00
Valor de sensación térmica. Posición 1 del Array: 43,06
Valor de salida final. Posición 1 del Array: 15,00
```

Ilustración 75 Menú 3-1

```
*********  RESULTADOS  *********

TEMPERATURA EXTERIOR

[21.65, 36.7]


HUMEDAD EXTERIOR

[45.09, 70.71]


SENSACION TERMICA

[18.82, 43.06]


TEMPERATURA LOCAL

[22.22, 7.57]


SALIDA FINAL

[20.0, 15.0]
```

Ilustración 76 Menú3-2

9.3.4 Ejecutar Arrays Semialeatorios

En esta sección el prototipo trabajará sin conexión a internet y realizará los cálculos simulando datos semialeatorios que en esta ocasión ya están determinados.

El sistema creará tres arrays de datos que estarán acotados dependiendo de los valores establecidos en el código y que en el manual del programador se mostrarán donde cambiarlo

Actualmente los valores establecidos son los siguientes:

- Temperatura exterior: Datos de 30° a 35° con un incremento de 2°.
    - Array con los siguientes datos: 30°-32°-34°
- Humedad exterior: Datos de 40% a 60% con un incremento de 5%.
    - Array con los siguientes datos: 40-45-50-55-60
- Temperatura local: Datos de 30° a 35° con un incremento de 2°.
    - Array con los siguientes datos: 30°-32°-34°

Esta intenta representar las variaciones de temperatura o humedad que se pueden dar para comprobar el resultado de los datos.

En la consola podremos ver lo siguiente según la configuración mencionada.

```
LISTA DATOS TEMPERATURA EXTERIOR
[30.0, 32.0, 34.0]
LISTA DATOS HUMEDAD EXTERIOR
[40.0, 45.0, 50.0, 55.0, 60.0]
LISTA DATOS TEMPERATURA LOCAL
[30.0, 32.0, 34.0]

********  RESULTADOS  ********

TEMPERATURA EXTERIOR
[30.0, 32.0, 34.0, 30.0, 32.0, 34.0, 30.0, 32.0, 34.0, 30.0, 32.0, 34.0, 30.0, 32.0, 34.0]

HUMEDAD EXTERIOR
[40.0, 40.0, 40.0, 45.0, 45.0, 45.0, 50.0, 50.0, 50.0, 55.0, 55.0, 55.0, 60.0, 60.0, 60.0]

SENSACION TERMICA
[33.0, 33.0, 33.0, 33.0, 33.0, 33.0, 33.0, 33.0, 33.0, 39.64, 40.14, 40.14, 43.32, 43.9, 43.9]


TEMPERATURA LOCAL
[30.0, 32.0, 34.0, 30.0, 32.0, 34.0, 30.0, 32.0, 34.0, 30.0, 32.0, 34.0, 30.0, 32.0, 34.0]


SALIDA FINAL
[38.69, 38.69, 38.69, 38.69, 38.69, 38.69, 38.69, 38.69, 38.69, 38.69, 38.69, 38.69, 38.69, 38.69, 38.69]
```

Ilustración 77 Menú 4

9.3.5 Ejecutar hilo con valores aleatorios

En esta sección del menú se ejecutará el prototipo en un hilo de manera ininterrumpida, pero esta vez con valores que no son reales, sino valores creados semialeatoriamente.

La creación de estos datos quiere simular el comportamiento real de la temperatura y humedad en un espacio acotado de tiempo.

Para ello los valores que se tratarán en el prototipo tienen una relación entre sí es decir, en primer lugar se ejecutará el prototipo con unos valores unitarios semialeatorios que estarán acotados de la siguiente manera:

- Temperatura local y exterior: de 20° a 35°.
- Humedad exterior: de 40 % a 100 %.

En las siguientes medidas se tomará este número como referencia y se calcularán los siguientes datos a partir del anterior es decir.

- En el caso de la temperatura cada dato obtenido sufrirá una subida o bajada de 2° respecto al anterior calculado.
- En el caso de la humedad cada dato obtenido sufrirá una subida o bajada de 5% respecto al anterior calculado.

La representación de los datos a través de la consola será la misma que se mostró en el primer punto de este apartado así como la representación de los Led's de la Raspberry Pi.

9.4 Manual del Programador

En este apartado se va a hacer una descripción de las pautas que se deben de tener en cuenta a la hora realizar algún cambio en el código del prototipo para mejorar su funcionamiento o cambiar alguna funcionalidad.

9.4.1 Lógica difusa

Para realizar cualquier cambio con respecto a las pautas tomadas en el proceso de trabajo de la lógica difusa establecida en el proyecto se deberán modificar los archivos contenidos en la carpeta *fcl* del proyecto

Modificar el archivo *signal.fcl* Para cualquier cambio con respecto al cálculo de la sensación térmica.

Modificar el archivo *signal.fcl* para cualquier cambio con respecto al cálculo de la sensación térmica.

Modificar el archivo *signal2.fcl* para cualquier cambio con respecto al cálculo de la salida final.

Dentro de los archivos, si se quieren modificar las variables lingüísticas de entrada, en los bloques *FUZZIFY* con el nombre de las variables se podrá hacer.

Para las variables de salida modificar el bloque *DEFUZZIFY*.

En cuanto a las reglas definidas están todas en los *RULEBLOCK* para su supervisión o modificación.

9.4.2 Peticiones HTTP y Paseador de datos

El parseo de datos que se realizó a la hora de consumir los datos obtenidos desde las redes de Internet de las cosas se encuentra en el paquete *com.httpREquest.*

En este paquete se puede encontrar dos clases, ambas se encargan de realizar la petición HTTP GET para recibir los datos de las distintas redes y de parsear los datos obtenidos.

Cada parseador y cada clase está programado para una URL en concreto, tener en cuenta que se si se desea consumir y parsear datos de otra fuente, lo más eficiente será desarrollar una nueva clase acomodada a las nuevas características.

Las clases se caracterizan por:

- HttpGetClass.java
    - Recogerá datos de la red de Internet de las cosas ThingSpeak cuya URL es: http://api.thingspeak.com/channels/17318/field/2.json .
- HttpGetClass2.java
    - Recogerá datos de la red de Internet de las cosas Midgar cuya URL es: http://156.35.82.104:3000/datos.json .

9.4.3 Configuración de PIN GPIO de la Raspberry.

La Raspberry Pi utiliza sus pines GPIO, a través de la librería *pi4j*, para la conexión en este caso con la placa a la que está acoplado el sensor de temperatura y el conversor MCP3008.

En el paquete *adc* del proyecto se encuentra la clase ADCObserver que es donde se definen la conexión exacta de los pines GPIO de la Raspberry Pi.

Si se va a utilizar otra librería u otro lenguaje de desarrollo como por ejemplo Python, estos pines GPIO tendrán otra nomenclatura apropiada al lenguaje o biblioteca que se use.

9.4.4 Clase principal JFuzzyClass.java

jFuzzyClass.java es la clase principal que se encuentra en el paquete *com.jFuzzy* desde donde la aplicación se ejecutará.

Se trata de la clase que ejecutará el menú mostrado del prototipo, dicha clase se encuentra correctamente documentada para lograr su funcionamiento, aún así se hará hincapié en varios apartados.

9.4.4.1 *Acotamiento de valores aleatorios del hilo (menú 5)*

Dentro del case del menú número 5 se encuentra el acotamiento inicial proporcionado para la ejecución de esta parte del menú.

Si se desea variar el acotamiento de los datos que se van a generar, las variables *temp, hum* y *tempLocal* son las que tomarán los valores iniciales para después seguir obteniendo datos a partir de ellas en el hilo que se lanzará a continuación.

En este mismo menú se puede cambiar la variación que se sufren los datos aleatorios posteriores creados.

La clase ThreadTempClass.java que instancia el menú cinco cuando crea un hilo de ejecución, contiene tres métodos llamados *temp(double), hum(double) y tempLocal(double)* que se encargan de generar los datos aleatorios siguientes y establecenel rango de variedad entre el dato generado y el dato a generar.

Actualmente son ±2° para las temperaturas y ±5% para la humedad.

9.4.4.2 *Definición de las URL donde se harán las peticiones HTTP*

El *case* 1 del menú que se encarga de la ejecución del prototipo con datos reales, instancia una clase llamada ThreadValoresReales.java a través de un hilo de ejecución donde define las llamadas a las URL de las plataformas de Internet de las cosas.

9.4.4.3 *Acotamiento de valores de Array semialeatorios (Menú 4)*

En el *case* 4 del menú se hace una llamada a un método *exeArrays ()* que cargará tres arrays de datos semialeatorios acotados.

Estos arrays se crearán a través de otros tres métodos llamados initArrayListTemp() , initArrayListHum() y initArrayListTempLocal().

En un principio esta acotación de valores simula la variación creciente tanto de temperatura como de humedad.

Accediendo a estos métodos podremos cambiar la acotación de los métodos y la variable de crecimiento entre los datos creados.

9.4.5 Sensor de temperatura

En el paquete *com.tmp36* se encuentra la clase *AnalogTemperatureSensorReader.java* que alberga la lectura de los datos del sensor, así como el algoritmo que se implementó para la calibración del sensor. En el caso de que se desee realizar cualquier otro cambio o calibración del sensor este será el lugar de cambios.

9.4.6 Control de los actuadores de la Raspberry Pi

En el paquete *com.tmp36* aparece otra clase, *GPIOTest.java* que al acceder a ella, se podrá observar en primer lugar la configuración de los pines GPIO que controlan los actuadores de la Raspberry Pi que en este caso serán los Led's.

Aquí también se podrá observar y modificar los usos y las acciones de los Led's con respecto a las lecturas de las salidas finales de lógica difusa.

Capítulo 10. Conclusiones y Ampliaciones

10.1 Conclusiones

10.1.1 Fi-Ware

Una de las principales conclusiones que a las que se ha llegado tiene que ver con el uso de la plataforma Fi-Ware, ya que fue en primer lugar era la plataforma elegida para el desarrollo de este proyecto fin de máster.

En un principio las aspiraciones de crear un proyecto válido con la plataforma Fi-Ware eran muy altas. Utilizando el conjunto de API´s que ofrecía así como otros recursos de los que ya se han hablado como el laboratorio on-line que se permite utilizar o la información que contienen las webs que hablan sobre esta plataforma.

Pero a medida que se fue investigando más en la plataforma más dudas ofrecía al respecto sobre el uso de la misma hasta que después de muchos errores que no me permitían avanzar se decidió cambiar de plataforma.

Como conclusión las aspiraciones que se tenía en un inicio eran demasiado altas y poco a poco fuera disminuyendo, no se considera que sea una plataforma que no se pueda utilizar sino que necesita una inversión de tiempo de investigación y desarrollo mucho más amplia de la que se disponía, así como que la comunidad de usuarios que la utilicen también crezcan para poder compartir más información y que el desarrollo de aplicaciones bajo esta plataforma sea más factible en un futuro.

10.1.2 Internet de las cosas.

Con respecto al cambio a la tecnología de Internet de las cosas, creo que para el desarrollo de este prototipo fue un gran acierto ya que con los pocos recursos que se disponía se pudo simular un entorno automatizado dotado de cierta inteligencia bastante fiable.

La facilidad que ofrecen las bibliotecas de Java con las que se trabajó, así como la gran ayuda de información que se puede obtener en la red de los distintos puntos en los que la plataforma se enfocó, hicieron que se pudiera desarrollar el prototipo sin grandes problemas que no permitieran un flujo de desarrollo normal incluso en el desarrollo de la instalación del Hardware adquirido sin tener muchos conceptos de electrónica, siendo la primera vez que existía contacto con elementos como los sensores, actuadores, placas o microcontroladores.

10.1.3 Lógica difusa

Por último una de las conclusiones a las que se llega viene enfocada en el desarrollo de aplicaciones utilizando lógica difusa. En el ámbito del que trata en el proyecto desarrollado está muy adaptado a la descripción del uso de lógica difusa. No se podría encontrar mejor ejemplo que el control de temperatura, ya que para empezar, la definición de los datos de temperatura hablando en términos de condiciones climáticas nunca va a estar definido en un solo punto, sino que con hemos visto, la mejor optimización es englobar variables lingüísticas que definan la totalidad del rango de variables físicas encontradas.

Respecto a este tema, se ha concluido un aspecto a mejorar en siguientes versiones y es que existen casos de comportamiento de datos tratados en bloques de lógica difusa, es decir en las salidas de los bloques de lógica difusa los valores obtenidos, no son valores exactos, sino como ya hemos visto, están establecidos en rangos de variables lingüísticas, lo que puede provocar que difiera bastante del número exacto obtenido, pero sigue estando en nuestra variable lingüística que es lo que se está buscando.

La solución de este aspecto, pasará por la definición de más variables lingüísticas acotando espacios de datos más reducidos. En el prototipo, cuando se trabaja con datos localizados en los extremos de los rangos de valores, es decir, temperaturas extremadamente altas o extremadamente bajas, los datos que se obtienen no se acercan a la realidad que queremos obtener con respecto a la sensación térmica entre dos valores de temperatura y humedad, pero ese valor no es lo que nos importa sino que lo que nos interesa es la variable lingüística a la que corresponde.

No importa que los valores extremos difieran entre sí de 10 o 20 grados ya que para nuestro sistema se considerado que a partir de un punto ese valor de temperatura es igual de extremadamente alto en 45 grados que en 65 grados, lo que importa es que si engloba la misma salida de variable lingüística que en este caso sería que la sensación térmica es extremadamente alta o extremadamente baja.

En la salida final del prototipo se ha encontrado un problema similar , ya que solamente se han definido cinco opciones de variables de salida, por lo que existen datos, sobre todo los que se encuentran en los extremos que albergan más cantidad de variables físicas. No obstante como se trata de un prototipo, queda bien definido que si la salida estuviese conectada a un termostato o aire acondicionado se podrían definir tantas salidas como se quisieran tener en cuenta.

10.2 Ampliaciones

10.2.1 Mejora del código

La ampliación más inmediata que se podrá realizar es una mejora del código, así como una refactorización de todas las clases y diseñar otra arquitectura de desarrollo para que en un futuro si se quieren adaptar sensores de la índole que sea, así como otras fuentes de consumo de datos, se puedan hacer de una manera más sencilla que la actual.

10.2.2 Interfaz Web gráfica

La creación de una interfaz Web y móvil gráfica para que el uso de la aplicación sobre el prototipo remotamente pudiera ser una ampliación de calidad y le ofrecería una funcionalidad al usuario muy completa.

Desde ver las condiciones atmosféricas de la sala o poner en marcha el dispositivo a distancia serían algunos de los servicios que se podrían desarrollar.

10.2.3 Big Data

Otra de las ampliaciones que se tuvo en cuenta fue la posibilidad de subir los todos los datos a una nube y una vez establecidos, analizarlos con Big Data para la obtener algunas conclusiones como en qué momentos se consume más energía, que condiciones son las ideales para el ahorro de energía o para establecer otros patrones de funcionamiento.

10.2.4 Mejora de lógica difusa

Como ya se ha visto y explicado la definición de la lógica difusa siempre puede ser más detallada y afinada según la cantidad de variables que se definan.

También se pueden cambiar las reglas y variables de la salida final del prototipo para obtener otro tipo de resultados que el mismo desarrollador quiera o incluso crear nuevos bloques de lógica difusa con otros datos como puede ser, número de personas que hay en el local o la luminosidad como factores para el control de temperatura

10.2.5 Otros sensores

Se podrán instalar otro tipo de sensores, mejorar los que existen, obtener otros datos como luminosidad y que se tomen esos datos como nuevas variables de otros paquetes de lógica difusa, replicar la instalación creada en varias habitaciones para controlar la temperatura de una casa entera interactuando con los distintos sensores de las habitaciones entre otros factores.

10.2.6 Crear canales de IOT propios

Crear canales propios en plataformas de Internet de las Cosas como pueden ser ThingSpeak, al igual que se hizo en Midgar y consumir los mismos datos creados y exponerlos para que la comunidad de datos abiertos de sensores sea más amplia.

Capítulo 11. Presupuesto

11.1 Desarrollo de Presupuesto Detallado

11.1.1 Recursos humanos

Categoría	Nº de horas	Precio €/hora	Precio Total
Programador/Investigador	1.232	7 €	8.624
Director del proyecto	208	20 €	4.160
Co-director del proyecto	1.040	15 €	15.600

Tabla 41 Presupuesto recursos humanos detallado

Estadísticas del proyecto

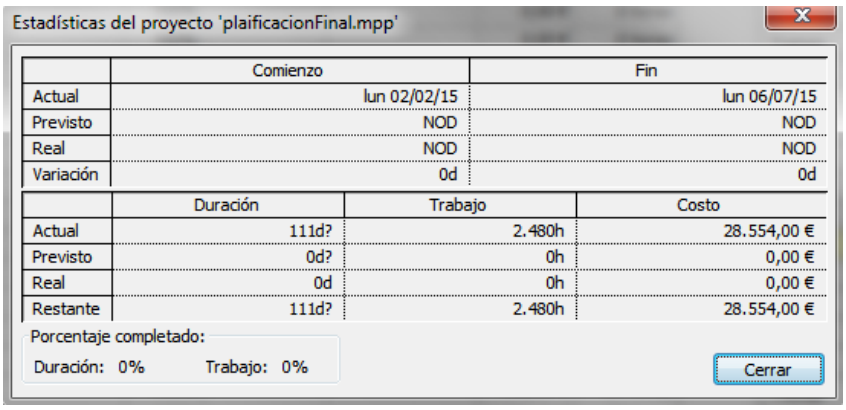


Ilustración 78 Resumen estadísticas del proyecto

11.1.2 Recursos para el desarrollo

- Se ha estimado una amortización con respecto a la vida útil de los recursos Hardware, que sería de unos 5 años.
- En el coste de los recursos software, se incluye la amortización de dichos productos. Para ello, se calculó aproximadamente una vida de 3 años de duración de los productos.
    - 3 años * 12 meses/año * 4 semanas/mes * 5 días laborables/semana = 720 días de uso.
    - Como se emplearon 111 días en la realización del proyecto, esto equivale a 111/480 = 0.24 → 24%
- La electricidad se calculó en base a las últimas facturas, de 70€ de media. Con la fórmula de 24 horas diarias durante 111 días.

Descripción	Unidades	Precio	Amortización	Importe

		Unitario		Amortizado
Ordenador Personal	1	600 €	20%	120 €
Windows 7 Profesional	1	200 €	20 %	40 €
Microsoft Office 2007 Professional	1	279 €	24 %	66,96 €
Enterprise Architect	1	199 €	24 %	47,76 €
Arduino	1	50 €	100 %	50 €
Raspberry	1	120 €	100 %	100 €
Conexión a Internet	1	45 €/mes	3.7 meses	166,5 €
Electricidad		0.10 € /día	111 días	11,1 €
Total				602,32 €

Tabla 42 Presupuesto Recursos Materiales

Presupuesto total

A cada categoría del presupuesto final se le ha añadido un suplemento del 7 % para posibles imprevistos que puedan ocurrir.

Categoría			Importe
Recursos Humanos			**28.554 €**
	Evaluación de Informes		**4.400 €**
		Presentación del Proyecto	1.056 €
		Informe de la tec. Fi-Ware	1.584 €
		Informe de la la tec. Azure	704 €
		Informe CMS CNN	528 €
		Elección proyecto	528 €
	Análisis		**4.752 €**
		Plataforma	1.584 €
		Fi-Ware	3.168 €
	Diseño		**6.864 €**
		Fi-Ware	6.864 €
	Evaluación de problemas		**4.320 €**
		Problemas tec. Fi-Ware	1.728 €
		Problemas integración GE	1.512 €
		Valoración cambio de tec.	1.080 €
	Rediseño del TFM		**8.218 €**
		Evaluación del prototipo	576 €
		Análisis	528 €
		Diseño	698 €
		Módulos	4.400 €
		Pruebas	672 €

	Documentación	1.344 €
Recursos materiales		**602,32 €**
Subtotal		29.156,32€
Imprevistos 5%		1.457,81€
Ganancias 20%		6.122,82€
IVA 21%		7.714,76€
TOTAL		44.415,27€

Tabla 43 Despliegue Presupuesto general

Capítulo 12. Referencias Bibliográficas

12.1 Libros y Artículos

[1] Atzori, L., Iera, A., Morabito, G., 2010. The Internet of Things: A survey. Comput. Networks 54, 2787–2805. doi:10.1016/j.comnet.2010.05.010

[2] González García, C., García-Bustelo, C.P., Espada, J.P., Cueva-Fernandez, G., 2014. Midgar: Generation of heterogeneous objects interconnecting applications. A Domain Specific Language proposal for Internet of Things scenarios. Comput. Networks 64, 143–158. doi:10.1016/j.comnet.2014.02.010

[3] Gonzalez Garcia, C., Espada, J.P., Nunez-Valdez, E.R., Diaz, V.G., 2014. Midgar: Domain-Specific Language to Generate Smart Objects for an Internet of Things Platform, in: 2014 Eighth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing. IEEE, Birmingham, United Kingdom, pp. 352–357. doi:10.1109/IMIS.2014.48

[4]Hribernik, K.A., Ghrairi, Z., Hans, C., Thoben, K., 2011. Co-creating the Internet of Things - First Experiences in the Participatory Design of Intelligent Products with Arduino, in: Concurrent Enterprising (ICE), 2011 17th International Conference on. Aachen, pp. 1–9.

[5] McFarlane, D., Sarma, S., Chirn, J.L., Wong, C.., Ashton, K., 2003. Auto ID systems and intelligent manufacturing control. Eng. Appl. Artif. Intell. 16, 365–376. doi:10.1016/S0952-1976(03)00077-0

[6] Meyer, G.G., Främling, K., Holmström, J., 2009. Intelligent Products: A survey. Comput. Ind. 60, 137–148. doi:10.1016/j.compind.2008.12.005

[7] LogMeIn. (2013). Xively. Retrieved June 23, 2013, from https://xively.com/

[8] Pintus, A., Carboni, D., Piras, A., 2011. The anatomy of a large scale social web for internet enabled objects. Proc. Second Int. Work. Web Things - WoT '11 1. doi:10.1145/1993966.1993975

[9] Pintus, A., Carboni, D., Piras, A., 2012a. Paraimpu: a platform for a social web of things, in: International World Wide Web Conference Com- Mittee (IW3C2). pp. 401–404.

[10] Pintus, A., Carboni, D., Piras, A., 2012b. Paraimpu [WWW Document]. https://www.paraimpu.com/. URL http://paraimpu.crs4.it/

[11] Piras, A., Carboni, D., Pintus, A., 2012a. A platform to collect, manage and share heterogeneous sensor data, in: 2012 Ninth International Conference on Networked Sensing (INSS). IEEE, Antwerp, pp. 1–2. doi:10.1109/INSS.2012.6240570

[12] Piras, A., Carboni, D., Pintus, A., Features, D.M.T., 2012b. A Platform to Collect , Manage and Share Heterogeneous Sensor Data, in: Networked Sensing Systems (INSS). Antwerp, pp. 1–2. doi:10.1109/INSS.2012.6240570

[13] Cueva-Fernandez, G., Pascual Espada, J., García-Díaz, V., Gonzalez-Crespo, R., 2015. Fuzzy decision method to improve the information exchange in a vehicle sensor tracking system. Appl. Soft Comput. 1–9. doi:10.1016/j.asoc.2015.01.066

[14**]** Acebal, César; Cueva Lovelle, Juan Manuel;"eXtreme Programming: un nuevo método de desarrollo de software". Novatica/Upgrade. 2002.

12.2 Referencias en Internet

http://www.loxone.com/eses/start.html **Loxone**

https://www.tado.com/es/ **Tado°**

http://ccia.ei.uvigo.es/docencia/IA/1213/transparencias/ejemplo-control-difuso.pdf **Trabajos teóricos lógica difusa**

http://www.laccei.org/LACCEI2007-Mexico/Papers%20PDF/IT228_AcevedoGauta.pdf **Trabajos teóricos lógica difusa**

https://www.google.es/url?sa=t&rct=j&q=&esrc=s&source=web&cd=5&cad=rja&uact=8&ved=0CDsQFjAE&url=http%3A%2F%2Frevcolfis.org%2Fojs%2Findex.php%2Frcf%2Farticle%2FdownloadSuppFile%2F420327%2F286&ei=vrKCVY_uBtGNsQSNqYHABQ&usg=AFQjCNFnmO7mzahOTUGxYd-ITrmlCLUxog&bvm=bv.96041959,d.cWc **Trabajos teóricos lógica difusa**

http://www.uhu.es/juanc.gutierrez/PID11030/PDFs/MercedesIsabel.pdf **Trabajos teóricos lógica difusa**

http://apc.io/ **VIA APC**

http://www.intel.es/content/www/es/es/do-it-yourself/edison.html **Intel Edison**

http://www.fiware.org/ **Fi-Ware**

http://www.esamur.com/jornadas/ponencias/ponencia91.pdf

https://campusvirtual.ull.es/ocw/pluginfile.php/2233/mod_resource/content/0/ApuntesCI.pdf **Teoría Lógica Difusa**

http://www.google.es/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0CCEQFjAA&url=http%3A%2F%2Fwww.revistas.ucr.ac.cr%2Findex.php%2Fcienciaytecnologia%2Farticle%2Fdownload%2F2640%2F2591&ei=fK2ZVfjEH8XaUYrsgbAH&usg=AFQjCNHPuexAZHH4nFYDhmCdvtI-AMyk6A&sig2=a2_0Z5z7ZL6AbMAKHhbB_g&bvm=bv.96952980,d.d24 **Teoría Lógica Difusa**

https://thingspeak.com/ **ThingSpeak**

https://xively.com/ **Xively**

https://www.paraimpu.com/ **Paraimpu**

https://www.arduino.cc/ **Arduino**

http://www.ajaxpro.info/ **JSON**

http://www.json.org/example.html **JSON**

https://www.processing.org/ **Processing**

http://wiring.org.co/ **Wiring**

https://www.java.com/es/ **Java**

http://jfuzzylogic.sourceforge.net/html/index.html **jFuzzy**

http://pi4j.com/ **Pi4j**

https://hc.apache.org/httpcomponents-client-ga/quickstart.html **Apache HTTP**

http://www.eclipse.org/home/index.php **Eclipse**

https://home.office.com/ **Office**

http://www.sparxsystems.com.au/products/ea/ **Enterprise Architec**

Capítulo 13. Apéndices

13.1 Glosario y Diccionario de Datos

.fcl: Archivo que contiene toda la configuración de las operaciones referidas a la lógica difusa.

Conjuntos difusos: conjunto con límites borrosos o "no muy bien" definidos.

Defuzzificacion: Conversión de las variables difusas de salida en un valor concreto de la variable real de salida empleando métodos matemáticos.

Fuzzificación: Conversión de los datos físicos en variables difusas mediante las funciones de pertenencia. Se asignan los grados de pertenencia correspondiente a cada una de las etiquetas lingüísticas definidas.

13.2 Contenido Entregado en el Archivo Adjunto

El archivo adjunto portará dos carpetas al descomprimirse, una de ellas llevará la carpeta *jFuzzy* que tiene lo referente al prototipo descrito.

Por otro lado encontramos una carpeta llamada *midgarapparduino* que mostrará el código fuente de una aplicación que sirve para recoger los datos de un sensor de temperatura instalado en Arduino.

13.2.1 Contenidos en carpeta jFuzzy

Se mostrará el contenido entregado en el archivo adjunto dentro de la carpeta jFuzzy

Directorio	Contenido
./ Directorio raíz de "desarrollo"	Contiene los ficheros de proyecto utilizado.
./doc	Contiene toda la documentación relativa al proyecto, incluyendo los ficheros generados por herramientas de generación de documentación automática como *Javadoc* o similar.
./fcl	Ficheros (.fcl) donde se encuentra parte de la programación de la lógica difusa.
./lib	Bibliotecas externas (*.jar*, *.dll*,…) necesarias para compilar y distribuir, de las que depende este proyecto.
./src	Ficheros fuente
./src/adc	Este directorio contiene los ficheros relacionados con el observador y los listeners desarrollados para la lectura de datos de los sensores.

./src/com	Contiene los ficheros relacionados con la aplicación principal
./src/com/httpRequest	Contiene los ficheros relacionados con las peticiones HTTP y el parseo de datos.
./src/com/jFuzzy	Contiene los ficheros relacionados con el funcionamiento principal del prototipo.
./src/com/test	Ficheron relacionados con las pruebas Junit.
./src/com/tmp36	Archivos relacionados con la lectura de datos del sensor tmp36 y el control de actuadores instalados en la Raspberry Pi.

13.2.2 Contenido en la carpeta midgarapparduino

Directorio	Contenido
./ Directorio raíz de "desarrollo"	Contiene los ficheros de proyecto utilizado.
./arduinoProgram	Programa desarrollado con el IDE Arduino que sirve para leer los datos de la placa y pasarlos a la aplicación.
./lib	Bibliotecas externas (*.jar*, *.dll*,…) necesarias para compilar y distribuir, de las que depende este proyecto.
./src	Ficheros fuente

13.3 Índice Alfabético

13.4 Código Fuente

A continuación, con respecto al código fuente se mostrarán solo los bloques de lógica difusa que se han usado.

13.4.1 Lógica difusa, sensación térmica

13.4.1.1 *Fichero "signal.fcl":*

```
   FUNCTION_BLOCK signal      // Block definition

VAR_INPUT                               // Define input variables
       temp_arduino : REAL;
       hum_thinkSpeak : REAL;
END_VAR

VAR_OUTPUT                              // Define output variable
       ext_cond : REAL;
END_VAR

FUZZIFY temp_arduino          // Fuzzify input variable 'temp_arduino'
       TERM extrmLow := TRIAN -15 -10 -5;
       TERM veryLow :=  TRIAN -7 -2 3;
       TERM low := TRIAN 0 5 15;
       TERM normal := TRIAN 12 18 24;
       TERM high := TRIAN 22 27 30;
       TERM veryHigh := TRIAN 28 33 38 ;
       TERM extremHigh := TRIAN 35 47 50;

END_FUZZIFY

FUZZIFY hum_thinkSpeak                // Fuzzify input variable 'hum_thinkSpeak'
```

```
        TERM veryLow :=  (0, 1) (20, 1) (40, 0);
        TERM low := TRIAN 30 45 60;
        TERM normal := TRIAN 50 60 70;
        TERM high := TRIAN 60 70 80;
        TERM veryHigh := (75, 0) (90, 1) (100,1) ;
END_FUZZIFY

DEFUZZIFY ext_cond                  // Defzzzify output variable 'signal'
        TERM extrmLow := TRIAN -15 -10 -5;
        TERM veryLow :=  TRIAN -7 -2 3;
        TERM low := TRIAN 0 5 18;
        TERM normal := TRIAN 14 18 24;
        TERM high := TRIAN 22 27 30;
        TERM veryHigh := TRIAN 28 33 38 ;
        TERM extremHigh := TRIAN 35 47 50;


        METHOD : COG;          // Use 'Center Of Gravity' defuzzification method
        DEFAULT := 0;          // Default value is 0 (if no rule activates defuzzifier)
END_DEFUZZIFY

RULEBLOCK No1

        AND : MIN;                    // Use 'min' for 'and' (also implicit use 'max' for
'or' to fulfill DeMorgan's Law)
        ACT : MIN;                    // Use 'min' activation method
        ACCU : MAX;                   // Use 'max' accumulation method


                                              //Reglas con respecto a la relación entre
temperatura y Humedad.


        RULE 1 : IF temp_arduino IS extrmLow AND hum_thinkSpeak IS veryLow THEN ext_cond
IS extrmLow;
        RULE 2 : IF temp_arduino IS extrmLow AND hum_thinkSpeak IS low THEN ext_cond IS
extrmLow;
        RULE 3 : IF temp_arduino IS extrmLow AND hum_thinkSpeak IS normal THEN ext_cond
IS extrmLow;
        RULE 4 : IF temp_arduino IS extrmLow AND hum_thinkSpeak IS high THEN ext_cond IS
veryLow;
        RULE 5 : IF temp_arduino IS extrmLow AND hum_thinkSpeak IS veryHigh THEN ext_cond
IS low;

        RULE 6 : IF temp_arduino IS veryLow AND hum_thinkSpeak IS veryLow THEN ext_cond
IS veryLow;
        RULE 7 : IF temp_arduino IS veryLow AND hum_thinkSpeak IS low THEN ext_cond IS
veryLow;
        RULE 8 : IF temp_arduino IS veryLow AND hum_thinkSpeak IS normal THEN ext_cond IS
veryLow;
        RULE 9 : IF temp_arduino IS veryLow AND hum_thinkSpeak IS high THEN ext_cond IS
low;
        RULE 10 : IF temp_arduino IS veryLow AND hum_thinkSpeak IS veryHigh THEN ext_cond
IS low;

        RULE 11 : IF temp_arduino IS low AND hum_thinkSpeak IS veryLow THEN ext_cond IS
veryLow;
        RULE 12 : IF temp_arduino IS low AND hum_thinkSpeak IS low THEN ext_cond IS low;
        RULE 13 : IF temp_arduino IS low AND hum_thinkSpeak IS normal THEN ext_cond IS
low;
        RULE 14 : IF temp_arduino IS low AND hum_thinkSpeak IS high THEN ext_cond IS low;
        RULE 15 : IF temp_arduino IS low AND hum_thinkSpeak IS veryHigh THEN ext_cond IS
normal;

        RULE 16 : IF temp_arduino IS normal AND hum_thinkSpeak IS veryLow THEN ext_cond
IS normal;
        RULE 17 : IF temp_arduino IS normal AND hum_thinkSpeak IS low THEN ext_cond IS
normal;
        RULE 18 : IF temp_arduino IS normal AND hum_thinkSpeak IS normal THEN ext_cond IS
normal;
        RULE 19 : IF temp_arduino IS normal AND hum_thinkSpeak IS high THEN ext_cond IS
normal;
        RULE 20 : IF temp_arduino IS normal AND hum_thinkSpeak IS veryHigh THEN ext_cond
IS normal;
```

```
        RULE 21 : IF temp_arduino IS high AND hum_thinkSpeak IS veryLow THEN ext_cond IS
normal;
        RULE 22 : IF temp_arduino IS high AND hum_thinkSpeak IS low THEN ext_cond IS
high;
        RULE 23 : IF temp_arduino IS high AND hum_thinkSpeak IS normal THEN ext_cond IS
veryHigh;
        RULE 24 : IF temp_arduino IS high AND hum_thinkSpeak IS high THEN ext_cond IS
veryHigh;
        RULE 25 : IF temp_arduino IS high AND hum_thinkSpeak IS veryHigh THEN ext_cond IS
extremHigh;

        RULE 26 : IF temp_arduino IS veryHigh AND hum_thinkSpeak IS veryLow THEN ext_cond
IS high;
        RULE 27 : IF temp_arduino IS veryHigh AND hum_thinkSpeak IS low THEN ext_cond IS
veryHigh;
        RULE 28 : IF temp_arduino IS veryHigh AND hum_thinkSpeak IS normal THEN ext_cond
IS extremHigh;
        RULE 29 : IF temp_arduino IS veryHigh AND hum_thinkSpeak IS high THEN ext_cond IS
extremHigh;
        RULE 30 : IF temp_arduino IS veryHigh AND hum_thinkSpeak IS veryHigh THEN
ext_cond IS extremHigh;

        RULE 31 : IF temp_arduino IS extremHigh AND hum_thinkSpeak IS veryLow THEN
ext_cond IS veryHigh;
        RULE 32 : IF temp_arduino IS extremHigh AND hum_thinkSpeak IS low THEN ext_cond
IS extremHigh;
        RULE 33 : IF temp_arduino IS extremHigh AND hum_thinkSpeak IS normal THEN
ext_cond IS extremHigh;
        RULE 34 : IF temp_arduino IS extremHigh AND hum_thinkSpeak IS high THEN ext_cond
IS extremHigh;
        RULE 35 : IF temp_arduino IS extremHigh AND hum_thinkSpeak IS veryHigh THEN
ext_cond IS extremHigh;

END_RULEBLOCK

END_FUNCTION_BLOCK
```

13.4.2 Lógica difusa, salida final

13.4.2.1 *Fichero "signal2.fcl":*

```
FUNCTION_BLOCK signal2

VAR_INPUT
        ext_cond : REAL;
        temp_local : REAL;
END_VAR

VAR_OUTPUT
        signal : REAL;
END_VAR

FUZZIFY ext_cond
        TERM extrmLow := TRIAN -15 -10 -5;
        TERM veryLow :=  TRIAN -7 -2 3;
        TERM low := TRIAN 0 5 15;
        TERM normal := TRIAN 12 18 24;
        TERM high := TRIAN 22 27 30;
        TERM veryHigh := TRIAN 28 33 38 ;
        TERM extremHigh := TRIAN 35 47 50;
END_FUZZIFY

FUZZIFY temp_local
        TERM extrmLow := TRIAN -15 -10 -5;
        TERM veryLow :=  TRIAN -7 -2 3;
        TERM low := TRIAN 0 5 15;
        TERM normal := TRIAN 12 18 24;
        TERM high := TRIAN 22 27 30;
        TERM veryHigh := TRIAN 28 33 38 ;
        TERM extremHigh := TRIAN 35 47 50;
END_FUZZIFY
```

```
DEFUZZIFY signal
       TERM on2 :=  (-15, 1) (10, 1) (15, 0);
       TERM on := TRIAN 10 15 20;
       TERM stable := TRIAN 18 20 22;
       TERM off := TRIAN 20 25 30;
       TERM off2 := (25, 0) (30, 1) (50,1) ;

       METHOD : COG;         // Use 'Center Of Gravity' defuzzification method
       DEFAULT := 0;         // Default value is 0 (if no rule activates defuzzifier)
END_DEFUZZIFY

RULEBLOCK No1


       AND : PROD;                  // Use 'min' for 'and' (also implicit use 'max' for
'or' to fulfill DeMorgan's Law)
       ACT : PROD;                  // Use 'min' activation method
       ACCU : SUM;                  // Use 'max' accumulation method

       RULE 1 : IF ext_cond IS extrmLow AND temp_local IS extrmLow THEN signal IS on2;
       RULE 2 : IF ext_cond IS extrmLow AND temp_local IS veryLow THEN signal IS on2;
       RULE 3 : IF ext_cond IS extrmLow AND temp_local IS low THEN signal IS on2;
       RULE 4 : IF ext_cond IS extrmLow AND temp_local IS normal THEN signal IS on;
       RULE 5 : IF ext_cond IS extrmLow AND temp_local IS high THEN signal IS stable;
       RULE 6 : IF ext_cond IS extrmLow AND temp_local IS veryHigh THEN signal IS off;
       RULE 7 : IF ext_cond IS extrmLow AND temp_local IS extremHigh THEN signal IS
off2;

       RULE 8 : IF ext_cond IS veryLow AND temp_local IS extrmLow THEN signal IS on2;
       RULE 9 : IF ext_cond IS veryLow AND temp_local IS veryLow THEN signal IS on2;
       RULE 10 : IF ext_cond IS veryLow AND temp_local IS low THEN signal IS on2;
       RULE 11 : IF ext_cond IS veryLow AND temp_local IS normal THEN signal IS on;
       RULE 12 : IF ext_cond IS veryLow AND temp_local IS high THEN signal IS off;
       RULE 13 : IF ext_cond IS veryLow AND temp_local IS veryHigh THEN signal IS off;
       RULE 14 : IF ext_cond IS veryLow AND temp_local IS extremHigh THEN signal IS
off2;

       RULE 15 : IF ext_cond IS low AND temp_local IS extrmLow THEN signal IS on2;
       RULE 16 : IF ext_cond IS low AND temp_local IS veryLow THEN signal IS on2;
       RULE 17 : IF ext_cond IS low AND temp_local IS low THEN signal IS on2;
       RULE 18 : IF ext_cond IS low AND temp_local IS normal THEN signal IS on;
       RULE 19 : IF ext_cond IS low AND temp_local IS high THEN signal IS stable;
       RULE 20 : IF ext_cond IS low AND temp_local IS veryHigh THEN signal IS off;
       RULE 21 : IF ext_cond IS low AND temp_local IS extremHigh THEN signal IS off2;

       RULE 22 : IF ext_cond IS normal AND temp_local IS extrmLow THEN signal IS on2;
       RULE 23 : IF ext_cond IS normal AND temp_local IS veryLow THEN signal IS on2;
       RULE 24 : IF ext_cond IS normal AND temp_local IS low THEN signal IS on;
       RULE 25 : IF ext_cond IS normal AND temp_local IS normal THEN signal IS stable;
       RULE 26 : IF ext_cond IS normal AND temp_local IS high THEN signal IS stable;
       RULE 27 : IF ext_cond IS normal AND temp_local IS veryHigh THEN signal IS off;
       RULE 28 : IF ext_cond IS normal AND temp_local IS extremHigh THEN signal IS off2;

       RULE 29 : IF ext_cond IS high AND temp_local IS extrmLow THEN signal IS on2;
       RULE 30 : IF ext_cond IS high AND temp_local IS veryLow THEN signal IS on2;
       RULE 31 : IF ext_cond IS high AND temp_local IS low THEN signal IS on;
       RULE 32 : IF ext_cond IS high AND temp_local IS normal THEN signal IS stable;
       RULE 33 : IF ext_cond IS high AND temp_local IS high THEN signal IS off;
       RULE 34 : IF ext_cond IS high AND temp_local IS veryHigh THEN signal IS off2;
       RULE 35 : IF ext_cond IS high AND temp_local IS extremHigh THEN signal IS off2;

       RULE 36 : IF ext_cond IS veryHigh AND temp_local IS extrmLow THEN signal IS on2;
       RULE 37 : IF ext_cond IS veryHigh AND temp_local IS veryLow THEN signal IS on2;
       RULE 38 : IF ext_cond IS veryHigh AND temp_local IS low THEN signal IS on;
       RULE 39 : IF ext_cond IS veryHigh AND temp_local IS normal THEN signal IS stable;
       RULE 40 : IF ext_cond IS veryHigh AND temp_local IS high THEN signal IS off;
       RULE 41 : IF ext_cond IS veryHigh AND temp_local IS veryHigh THEN signal IS off2;
       RULE 42 : IF ext_cond IS veryHigh AND temp_local IS extremHigh THEN signal IS
off2;

       RULE 43 : IF ext_cond IS extremHigh AND temp_local IS extrmLow THEN signal IS
on2;
       RULE 44 : IF ext_cond IS extremHigh AND temp_local IS veryLow THEN signal IS on2;
       RULE 45 : IF ext_cond IS extremHigh AND temp_local IS low THEN signal IS on;
       RULE 46 : IF ext_cond IS extremHigh AND temp_local IS normal THEN signal IS off;
```

```
        RULE 47 : IF ext_cond IS extremHigh AND temp_local IS high THEN signal IS off2;
        RULE 48 : IF ext_cond IS extremHigh AND temp_local IS veryHigh THEN signal IS
off2;
        RULE 49 : IF ext_cond IS extremHigh AND temp_local IS extremHigh THEN signal IS
off2;



END_RULEBLOCK

END_FUNCTION_BLOCK
```