E.T.S. de Ingeniería Industrial,
Informática y de Telecomunicación

Desarrollo de una aplicación Android para la eficiencia energética en instalaciones KNX

Grado en Ingeniería
en Tecnologías de Telecomunicación

Trabajo Fin de Grado

Adrián Iglesias Martín

Ignacio R. Matías Maestro

Ander Gabilondo Areta

Pamplona, 1 de Julio 2016

upna Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

Agradecimientos

He de agradecer fundamentalmente a Ander Gabilondo Areta el poder haber realizado este trabajo, él ha sido quien me lo ha propuesto y guiado en su elaboración. También he de agradecerle el haberme facilitado el material necesario para su realización y me ha dado la información necesaria para el manejo de dichos materiales.

Lista de palabras clave:

- Eficiencia Energética
- KNX
- Sensores
- Raspberry Pi
- Java
- Android
- Android Studio
- Base de datos
- Gráficas

Índice

Parte 1: Introducción al trabajo

1.1 Resumen

El presente trabajo trata del desarrollo de una aplicación software capaz de recibir datos de una instalación domótica y medir la eficiencia energética de ésta.

Dicha instalación consta de:

- **Un sensor de temperatura** encargado de medir la temperatura interior de la instalación.

- **Una sonda de temperatura** que medirá la temperatura exterior de la instalación.

- **Un sensor de consumo energético** que medirá el consumo de la instalación.

Los datos obtenidos por estos sensores son enviados a un ordenador de placa reducida llamado Raspberry Pi. Para hacer dicho envío se usará un dispositivo KNX intermediario (BAOS_KNX module 820) que hará que las tramas KNX puedan llegar a los puertos serie de la Raspberry Pi. La Raspberry analizará las tramas recibidas mediante código Java que estará corriendo en ella y las almacenará en una base de datos de un dominio web para que los datos puedan ser accesibles por otros dispositivos.

Finalmente se ha desarrollado una aplicación Android para recoger los datos de la base de datos y manipularlos para mostrar de forma gráfica los resultados y que se pueda apreciar de forma sencilla la eficiencia de la instalación.

1.2 Objetivos

El objetivo de este trabajo es medir la eficiencia energética térmica de una instalación y mostrar los datos obtenidos en gráficas mediante una aplicación Android, relacionando el consumo con la normalización de eficiencia (consumo/grados-día).

Hoy en día existen distintos tipos de sistemas que permiten medir la eficiencia energética de cualquier instalación, debido a esto, este trabajo está pensado para que los datos obtenidos y el resultado final sean más visuales para el usuario.

El presente trabajo está desarrollado en tecnología KNX y los datos obtenidos son de la temperatura de una instalación, tanto en su interior como en su exterior, además de los datos de consumo del sistema.

Cabe decir que este proyecto sólo está orientado a medir la eficiencia energética térmica pero que también se podrían medir otros tipos de eficiencia, tales como: lumínica, humedad, etc.

1.3 Introducción

En los últimos años el uso de la tecnología va aumentado de forma considerable. Tal está siendo alcance que hoy en día, la gran mayoría de las nuevas instalaciones llevan en su interior cierto sistema domótico, que contribuye a ahorrar en gastos y a hacer más cómoda la estancia a las personas.

Las actuales instalaciones constan de varios sensores que miden temperatura, luminosidad, humedad... actuando según los datos que se recogen de ellos. Por ejemplo: apagando una luz si hay mucha luminosidad o encendiendo la calefacción si la temperatura ambiente está por debajo de los límites.

Se prevé que en un futuro toda instalación contará con sistemas domóticos para ahorrar en gastos innecesarios y demostrar lo eficiente que puede llegar a ser.

Por otro lado la mayoría de la población mundial tiene en su poder un teléfono móvil o Smartphone. Dado esto, la mayoría de empresas domóticas dan un servicio a sus usuarios con el que pueden acceder a sus instalaciones, para ver el estado y actuar sobre ellas con dispositivos móviles. De manera que el usuario final tiene poder absoluto sobre su instalación, y puede modificarla a su antojo o ver su eficiencia independientemente de si está en ella o no.

Este trabajo está orientado a esto último, se recogen datos de una instalación para después poder ser observados y evaluados de una manera gráfica a través de un móvil, Tablet, smartTV,…

Parte 2: Estado del Arte

En esta parte se va a proceder a describir los distintos programas, protocolos, tecnologías y dispositivos que se han utilizado para llevar a cabo el trabajo, ordenados cronológicamente según su uso en el proyecto, es decir, primero aparecerán los elementos necesarios en un inicio y por último los elementos finales.

2.1 KNX

Figura 1: Logo KNX

Según la página web oficial:

"KNX es un estándar mundial para la automatización de viviendas y edificios. Un estándar en el sector de la_domótica, inmótica y urbótica, una tecnología fiable, una forma de comunicación totalmente transparente a fabricantes, marcas, diseños y arquitecturas. Todo con el único fin de ofrecer al usuario final los tres pilares básicos de la domótica:

- Confort
- Seguridad
- Eficiencia energética

Es un protocolo estándar porque todos los elementos que intervienen en la instalación utilizan un protocolo común para comunicarse.

La tecnología KNX está respaldada por la KNX Association, un grupo de compañías líderes en el mercado activas en muchas áreas de aplicación relativas al control de casas y edificios. Actualmente, la KNX Association tiene más de 100 empresas miembros, éstas engloban más del 80% de los dispositivos vendidos en Europa para el control de casas y edificios. Como meta común, estas compañías promueven el desarrollo de sistemas de control de casas y edificios en general, así como también promueven KNX como el único estándar abierto mundial para el control de casas y edificios. A nivel mundial, la KNX Association tiene acuerdos partners con más de 21.000 compañías integradoras en 70 países, más de 50 universidades técnicas, así como más de 100 centros de formación."[1]

La especificación KNX fue publicada en 2002 por KNX Association, después de que en 1997 los tres grandes consorcios que manejaban las soluciones para el control de viviendas y edificios en Europa; Batibus(Francia, Italia, España), EIB(Alemania y norte de Europa) y EHS(productos de línea blanca y marrón), decidieran unir fuerzas creando un único estándar de edificios inteligentes para todo Europa y acordando crear una norma industrial común que también podría ser propuesta como norma internacional.

A continuación se mencionan las ventajas de usar KNX según su página web:

- "Considerable ahorro energético, resultado del bajo coste operacional: La iluminación y la calefacción sólo se encenderán cuando sea necesario dependiendo de los perfiles seleccionados y/o de la presencia en la habitación, de esta forma ahorrará energía y dinero. Lo que es más, la iluminación puede ser controlada automáticamente según la intensidad de luminancia del día, manteniendo un nivel mínimo de claridad en cada lugar de trabajo y reduciendo así el consumo energético (sólo permanecerán encendidas aquellas luces que realmente necesite).
- Ahorro de tiempo: Enlazar todos los dispositivos de control sencillamente mediante un bus, reduce considerable el tiempo de diseño e instalación. Una única herramienta que es independiente de la aplicación y del fabricante llamada "Engineering Tool Software" (ETS) permiten el diseño, la implementación y la configuración de la instalación que posea productos certificados KNX. Debido a que la herramienta es independiente del fabricante, el integrador de sistemas podrá combinar en la instalación los productos de diferentes fabricantes en diferentes medios de comunicación (par trenzado, radio frecuencia, línea de fuerza y/o IP/Ethernet).
- Escalabilidad y facilidad a mejoras futuras: Una instalación KNX puede ser fácilmente adaptada y ampliada a nuevas aplicaciones. Los nuevos componentes se pueden conectar directamente a la instalación bus existente."[1]

KNX tiene una tecnología flexible y distintos medios de transmisión:

- TP (Par trenzado)
- PL110 (Powerline, red eléctrica)
- KNX-RF (Radio frecuencia)
- Ethernet (KNXIP)

En este proyecto la tecnología que utilizaremos será la TP sobre par trenzado a 9600bps.

El hardware de KNX consta básicamente de 4 grupos de elementos:

- **Actuadores**: Son los elementos que se conectan físicamente con los dispositivos sobre los que se quiera actuar en la instalación. Luces, motores, etc.
- **Sensores**: Los sensores son los elementos del sistema que recogen datos o interpretan órdenes del usuario.

- **Pasarelas**: Enlazan otros sistemas con otros protocolos de comunicación con KNX. LONWORKS, X10 etc. a KNX.
- **Acopladores**: Son los encargados de la separación física del bus, pudiendo hacer división de áreas, grupos y líneas.

En lo que al software se refiere KNX distingue 2 tipos de software:

- **Software de gestión**: El software de gestión, es decir el que se usará para configurar los dispositivos y ponerlos en marcha es el ETS que explicaremos en el apartado 2.2. Esta herramienta es la única forma de configurar los dispositivos KNX y es creada, suministrada y regulada únicamente por la KNX Association.
- **Software de control**: Es el programa que permite tener control de la instalación y actuar sobre ella con distintas funciones.

En este trabajo se ha optado por trabajar con dispositivos KNX debido a que es el estándar más usado en Europa. Esto hace que la aplicación pueda ser incorporada en más instalaciones que usando otros protocolos y poder conseguir sus dispositivos sea más sencillo.

2.2 ETS5

Figura 2: Log ETS5

A continuación, se mostrará la descripción que hace KNX de su herramienta software:

"Las siglas ETS significan Engineering Tool Software (herramienta de software de ingeniería). Se trata de una herramienta independiente de cualquier fabricante y sirve para diseñar y configurar instalaciones inteligentes para el control de viviendas y edificios basadas en KNX. El software ETS funciona en ordenadores con sistema operativo Windows y fue comercializado por primera vez en 1993.

La KNX Association como fundadora y propietaria del estándar KNX ofrece con el ETS una herramienta que de hecho es parte del propio estándar, y en consecuencia también parte del sistema KNX. Ello implica varias ventajas importantes:

- Garantía de máxima compatibilidad entre software ETS y estándar KNX.
- Todas las bases de datos de productos certificados de todos los fabricantes KNX pueden ser importados al ETS.

- Compatibilidad del ETS con versiones anteriores (hasta ETS2) en respecto a datos de productos y proyectos respalda sus resultados de trabajo y permiten editarlos.
- Todos los integradores e ingenierías en cualquier parte del mundo usan una única herramienta para todos los proyectos y todos los productos certificados: ello garantiza un intercambio de datos seguro."[2]

Como ya se ha explicado anteriormente el sistema de este trabajo consta de dispositivos KNX por lo que es esencial el uso de este programa para poder configurar dichos dispositivos.

2.3 Dispositivos KNX

Figura 3: Sensor de temperatura KNX ABB

Figura 4: Sensor de consumo Zennio KNX

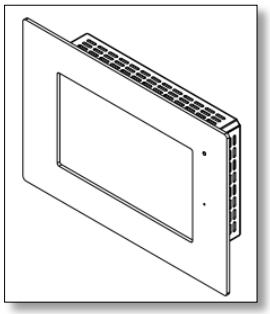


Figura 5: Pantalla KNX HC2

Figura 6: Sonda de temperatura DW-TS-N1P

Figura 7: Fuente de alimentación KNX

Los sensores utilizados en este trabajo utilizan el protocolo de comunicación KNX anteriormente explicado, igual que la pantalla que recoge los datos de la sonda.

Para la realización del proyecto son necesarios: 2 sensores y una sonda de temperatura conectada a una pantalla con bus KNX. Esta sonda recogerá los datos de la temperatura exterior de la instalación y los enviará a la pantalla KNX que se encargará de mandar dichos datos al bus KNX. Uno de los sensores será un sensor de temperatura que recogerá los datos de temperatura interior de la instalación. Y el otro sensor se encargará de medir el consumo energético de la instalación.

Todos estos dispositivos comentados estarán conectados a una fuente de alimentación KNX.

2.4 KNX_BAOS module 820

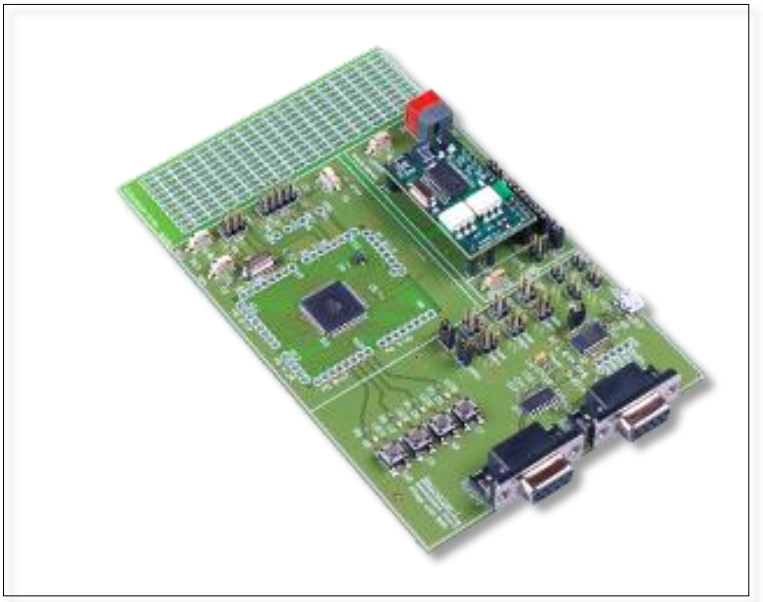


Figura 8: Placa KNX_BAOS module 820

La placa pequeña sobresaliente de la figura 8 tiene unas dimensiones de 44x25 mm. Esta placa consta de puertos serie de 5V DC y puertos bus KNX +/-, con un consumo de corriente de 9mA.

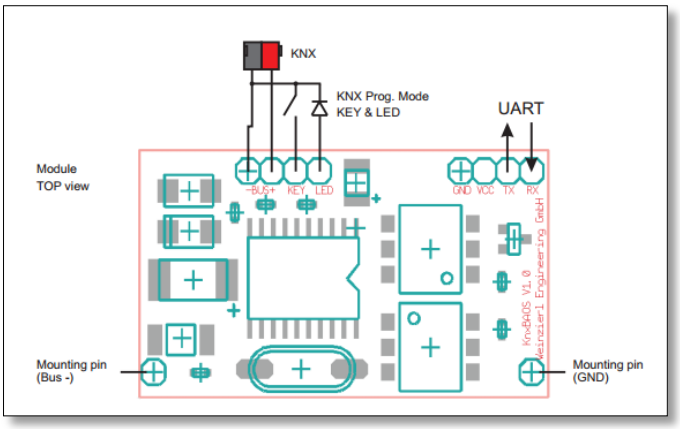


Figura 9: Placa KNX_BAOS esquema de puertos

La placa convertirá los distintos voltajes del bus KNX en un voltaje en DC de 5V para los puertos UART y viceversa. Gracias a esto se hace que la Raspberry Pi pueda recibir las tramas KNX por sus puertos UART (Universal Asynchronous Receiver Transmitter), ya que estos se conectarán con los puertos UART del módulo, a la vez que se conectarán los puertos bus KNX al bus KNX. Cabe destacar que la placa tiene que ser alimentada por corriente para su correcto funcionamiento.

La placa genera su propia trama cuando detecta flujo por ella según el anexo 1. Esto nos hace poder crear programas Java que detecten el inicio de dichas tramas.

Las tramas generadas al recibir datos de forma asíncrona son de la siguiente manera:

Figura 10: Estructura de la cabecera de las tramas

Una cabecera de 4 bytes en la cual el primero es un byte con valor 0x68 los que nos permitirá solo capturan la trama que nos interesa. Los siguientes 2 bytes hacen referencia a la longitud de la trama. En este trabajo al tener 2 tipos de datos recibidos (2 bytes o 4 bytes) tendremos tramas de 9 u 11 bytes después de la cabecera, siendo el valor de estos 0x09,0x09 en el caso de datos de 2 bytes o 0x0B,0x0B en el caso de datos 4 bytes. El último byte que cierra esta cabecera será el mismo que el que la abre es decir el 0x68.

El byte anterior a la trama lo genera KNX como se puede apreciar en el anexo1.

Los primeros 2 bytes de la trama son 0xF0 y 0xC1 que son el comienzo de una trama que se recibe asíncronamente con el dato enviado.

Offset	Field	Size	Value	Description
+0	MainService	1	0xF0	Main service code
+1	SubService	1	0xC1	Subservice code
+2	StartDatapoint	1		ID of first datapoint
+3	NumberOfDatapoints	1		Number of datapoints in this indication
+4	First DP ID	1		ID of first datapoint
+5	First DP state/length	1		State/length byte of first datapoint
+6	First DP value	1-14		Value of first datapoint
...	...		...	...
+N-2	Last DP ID	1		ID of last datapoint
+N-1	Last DP state/length	1		State/length byte of last datapoint
+N	Last DP value	1-14		Value of last datapoint

Figura 11: Estructura de las tramas

Después de estos bytes viene el byte que indica el Id de la trama, es decir de que te dice de que dispositivo KNX viene. En este caso podrán ser de 3 valores diferentes; 0x02 si los datos provienen del sensor de temperatura en el interior de la instalación, 0x03 si proviene del sensor de temperatura del exterior de la instalación y 0x04 si los datos llegan del sensor de consumo. El próximo byte indica el número de datos que contiene la trama, en este proyecto solo recibimos un dato por cada trama así que su valor será de 0x01, si no es así se descartará la trama. El siguiente byte contendrá el Id del primer dato de la trama por lo que será del mismo valor que en el caso del Id anterior.

El siguiente byte es el valor del tamaño de los datos 0x82 si es de 2 bytes y 0x84 si es de 4 bytes. Los siguientes 2 o 4 bytes ya serán el valor de los datos recibidos y los últimos bytes de la trama serán el checksum de la trama. Cabe destacar que el primer bit de los bytes de datos de 4 bytes indica el signo del dato, 0 si es positivo y 1 si es negativo.

Trama de 2 bytes del sensor de temperatura interior y con valor 15:

68-09-09-68-D3-F0-C1-02-01-02-82-0F-00

Trama de 2 bytes del sensor de temperatura exterior y con valor 15:

68-09-09-68-D3-F0-C1-03-01-03-82-0F-00

Trama de 4 bytes del sensor de consumo con valor 8500:

68-0B-0B-68-D3-F0-C1-04-01-04-84-00-00-21-34

2.5 Raspberry Pi

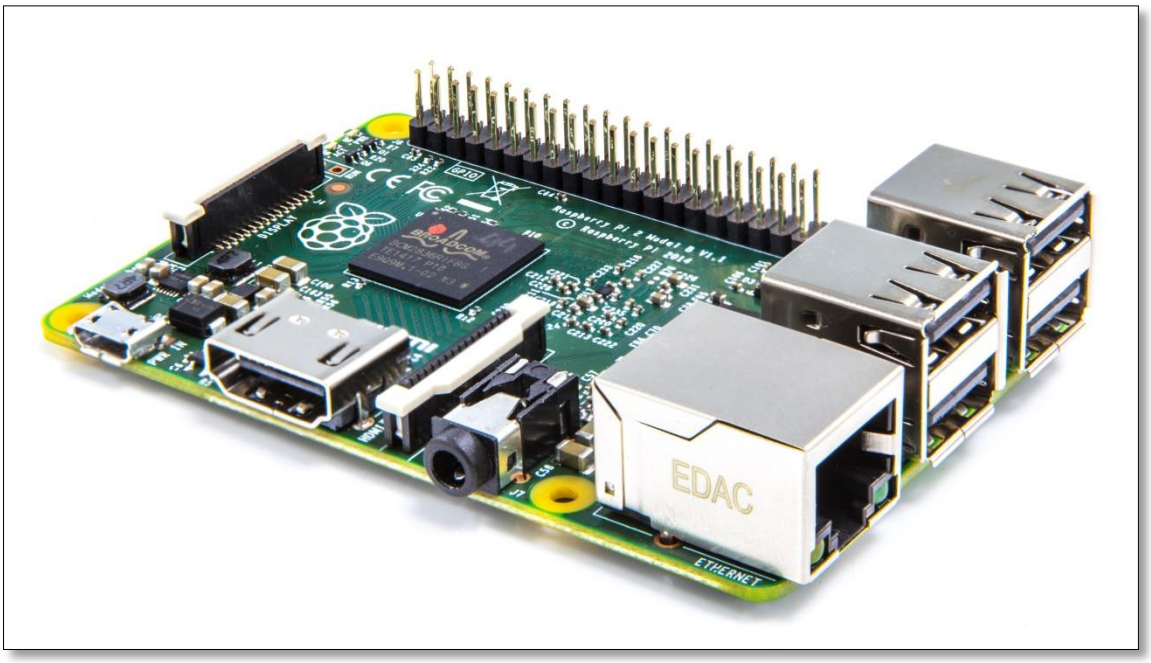


Figura 12: Raspberry Pi

Raspberry Pi es un ordenador de placa reducida (85.6 x 56.5 mm) y de bajo coste desarrollada en Reino Unido por la Fundación Raspberry Pi. Se entiende que es un producto con propiedad registrada pero de uso libre. En cuanto el software es open source, teniendo un sistema operativo oficial como es Raspbian, que es una adaptación del sistema operativo Debian. Aunque permite cualquier otro sistema operativo como Windows, Linux, Android, etc. En nuestro caso el sistema operativo utilizado es el oficial Raspbian.

En cuanto al hardware se distinguen 2 tipos de placas: una es el modelo A y la otra el modelo B. Las características que distinguen ambas placas son, por un lado, que el modelo A únicamente tiene una salida USB mientras el modelo B tiene 4 y por otro lado, una característica fundamental, que el modelo B tiene un puerto RJ45 que permite la conexión Ethernet.

En este trabajo se ha hecho uso del modelo B de la Raspberry Pi ya que se necesita conectar la Raspberry a una red Ethernet para hacer el envío de los datos.

Para la realización de este proyecto se ha usado los siguientes componentes de la Raspberry:

- Puerto RJ45: Para conectar la Raspberry a internet.
- Puertos UART, rxtx GPIO: Se ha utilizado para conectar la Raspberry a la placa BAOS y recibir los datos del bus KNX para su posterior procesado.

- Puerto HDMI: Sirve para ver el interfaz gráfico de la Raspberry y poder programarla a nuestro gusto.
- Puertos USB: Se ha utilizado para conectar otros componentes hardware para poder programar en ella, tales como, un ratón y un teclado.
- Ranura microSD: Sirve para introducir la tarjeta microSD con el sistema operativo que se le quiera poner a la Raspberry Pi.
- Micro USB(power In): Es el puerto de alimentación de la placa con el cual ponemos en funcionamiento la Raspberry.

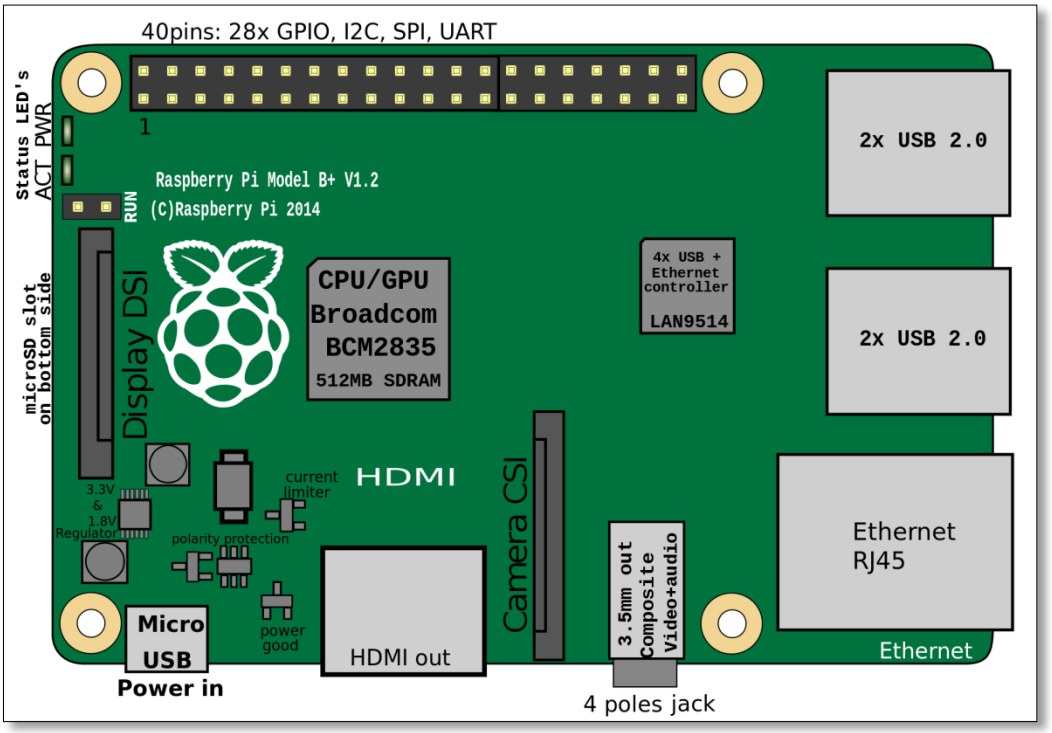


Figura 13: Raspberry Pi Esquema del sus puertos de salida y entrada

La Raspberry es uno de los elementos fundamentales de este trabajo ya que es el elemento que se comunica con los dispositivos KNX, más concretamente con la placa BAOS, y recoge los datos que viajan por el bus KNX mediante un programa JAVA que pone a escuchar los puertos UART y almacena todo lo recibido en un base de datos.

Cabe destacar que los puertos serie o UART de la Raspberry son los pines 8 y 10 de la placa de pines.

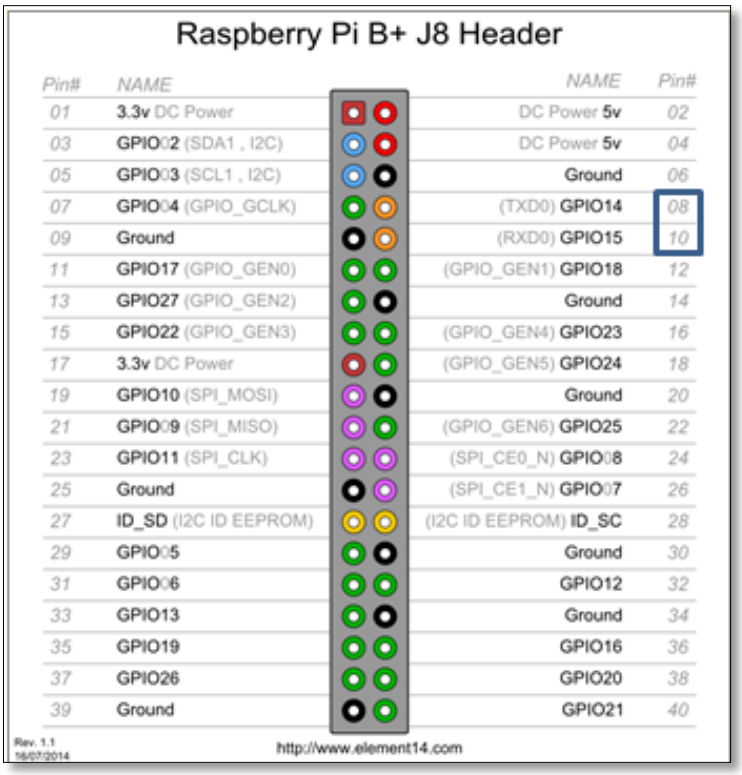


Figura 14: Pines de la Raspberry Pi

2.6 Java

Figura 15: Logo Java

Java es un lenguaje de programación de propósito general y orientado a objetos que fue comercializado por primera vez en 1995. En este trabajo se ha usado una versión Java compatible con un procesador ARM que es el utilizado en la Raspberry Pi.

Java permite el uso de librerías externas, lo que facilita el programar, ya que puedes usar aquellas útiles para tu proyecto e introducirlas en él rebajando el tamaño del código final y ofreciéndote más opciones de programado.

Cabe destacar que se ha usado Java para recoger los datos que llegan a la Raspberry, procesarlos y almacenarlos en una base de datos situada en un dominio web. Para realizar el almacenamiento creamos objetos JSON con Java y se realiza una petición

web a un PHP situado en el dominio web. El PHP se encargará de almacenar los datos que le lleguen a la base de datos del dominio.

2.7 PHP y MySQL

Figura 16: Logo PHP

Figura 17: Logo MySQL

PHP es otro lenguaje de programación de uso general y orientado más al lado del servidor, originalmente diseñado para el desarrollo web de contenido dinámico.

MySQL en cambio, es un sistema de gestión de base de datos muy utilizado en aplicaciones web. Normalmente MySQL aparece ligada a PHP y se maneja la administración mediante la herramienta phpMyAdmin.

En este trabajo se ha usado: 2 PHP, situados en un dominio web el cual contiene una base de datos MySQL y la herramienta phpMyAdmin. Uno de los PHP se encargará de guardar todo lo recibido en la base de datos y el otro se encargará de recoger todos los datos almacenados en la base de datos para codificarlos como JSON y enviarlos al dispositivo Android que requiera de los datos de la base de datos.

2.8 Android

Figura 18: Logo Android

Android es el sistema operativo implementado en la mayoría de los dispositivos móviles actualmente. Este será el sistema operativo en el que instalaremos la aplicación para poder ver el resultado final del trabajo con la finalidad de que cualquier persona pueda ver lo que sucede en su instalación desde cualquier lugar. Este sistema permite programar aplicaciones en una variación de Java llamada Dalvik.

El sistema operativo también proporciona todas las interfaces necesarias para poder desarrollar aplicaciones que puedan acceder a las funciones del teléfono (agenda, registros, etc.).

2.9 Android Studio

Figura 19: Logo Android Studio

Android Studio es un entorno de desarrollo integrado (IDE), basado en IntelliJ IDEA de la compañía JetBrains. Android Studio utiliza una licencia de software libre Apache 2.0, está programado en Java y es multiplataforma.

Android Studio es una herramienta muy útil para crear aplicaciones Android debido a todas sus opciones. Te permite de forma sencilla generar el contenido para tu aplicación y ver una vista previa de cómo será esta una vez ejecutada. En Android Studio se trabaja con Layouts, pestañas que simulan la pantalla de un dispositivo móvil. Sobre estos layouts se pueden insertar distintos tipos de objetos como botones, editores de texto, etc.

Todo layout se puede relacionar con una clase java para aplicar distintas funciones a los layouts y sea más interactivo para los usuarios. Aparte de los layouts y clases java Android Studio también consta de un manifiesto donde deberemos declarar las distintas clases java y un build.gradle donde se indica la versión Android en la que se quiera programar y las dependencias o librerías externas que tiene la aplicación.

Android Studio también te permite ejecutar la aplicación en el móvil, si este está conectado al ordenador en el que estés trabajando, mediante un USB. De esta forma se irá modificando el código sin necesidad de generar todo el rato una apk e instalándola en el móvil.

En este trabajo toda la parte de la aplicación Android se ha generado con esta herramienta. Debido a que su programación se basa en Java ha sido muy sencillo importar librerías externas las cuales han facilitado: la realización de las gráficas, hacer las peticiones web y decodificar los objetos JSON que recibimos al hacer las llamadas al PHP.

2.10 Esquema del Montaje final de los anteriores elementos

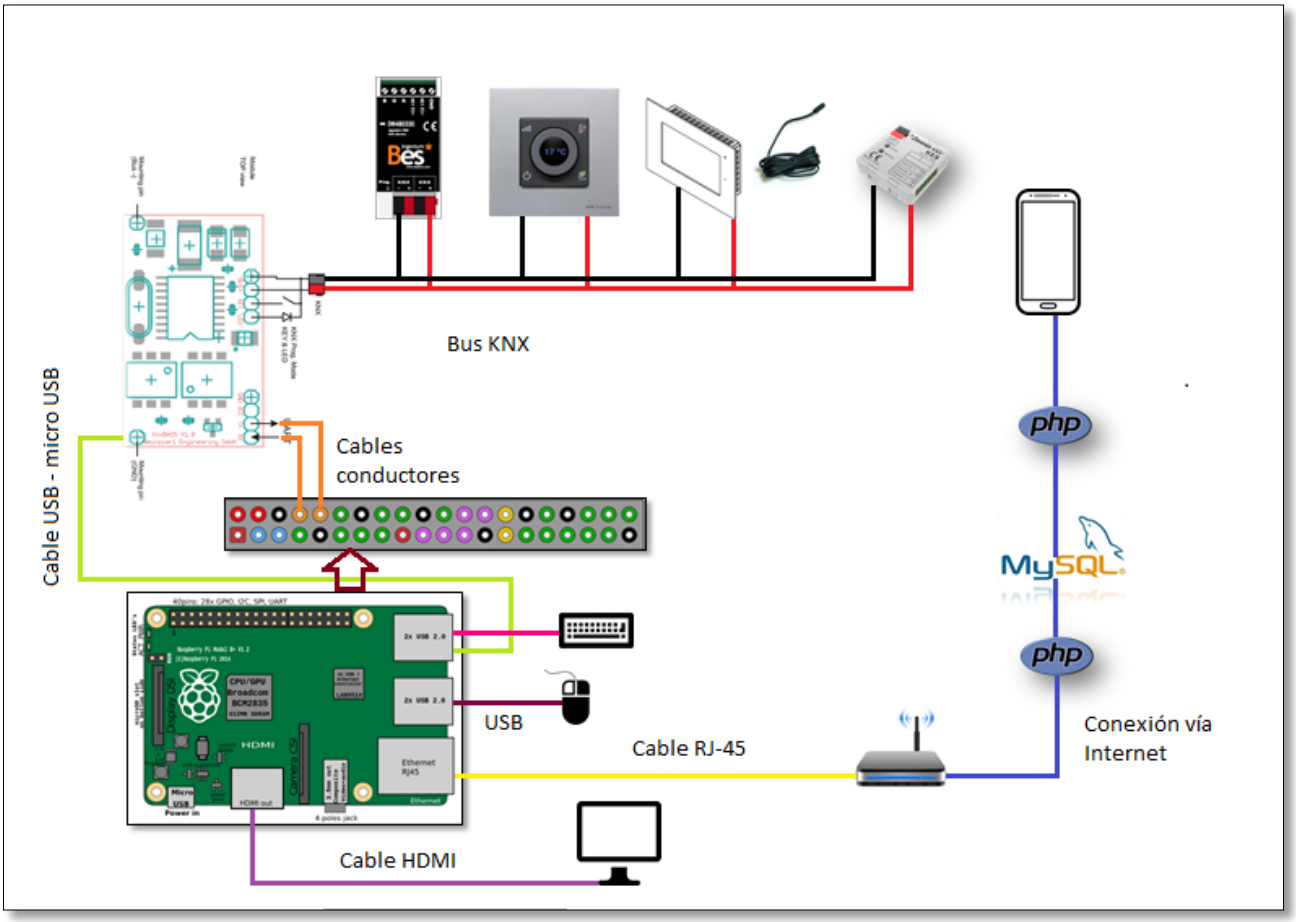


Figura 20: Esquema del montaje final

Parte 3: Desarrollo técnico

Se va a proceder a describir parte por parte como se han realizado todas las fases del trabajo de una manera técnica.

Este trabajo se puede dividir en tres grandes apartados, seccionado por los dispositivos con los que se trabaja en cada apartado y sus correspondientes lenguajes de programación.

La primera sección, es la que se encarga de programar y conectar los distintos dispositivos KNX y hacer la conexión con la Raspberry.

La segunda sección, es la que se encarga de programar la Raspberry y hacer los códigos Java que correrán en ella para recibir las tramas y su posterior envió a la base de datos. En esta parte también se explicarán los códigos PHP y la tabla de datos que almacena todos los datos.

Finalmente la última sección, es la referente a los dispositivos móviles, es decir, en ella se explicará cómo se ha desarrollado la aplicación Android con el lenguaje de programación Java Android y el IDE Android Studio.

3.1 Parte 1: Programación y conexión de los dispositivos KNX

En esta parte se va a describir como se ha conectado y programado todos los dispositivos KNX.

3.1.1 Conexión de los dispositivos

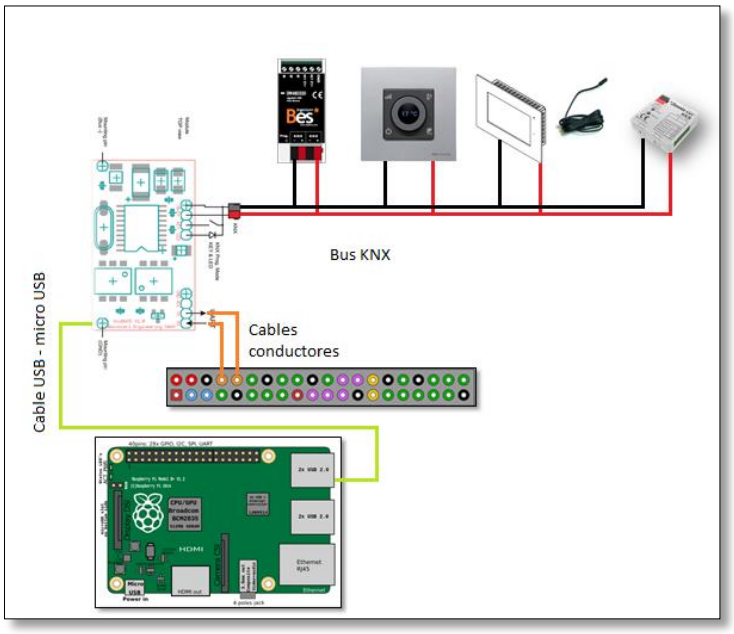


Figura 21: Conexionado de los dispositivos KNX y la Raspberry Pi

Todos los dispositivos KNX van conectados a la fuente de alimentación KNX mediante cables metálicos. La placa KNX_BAOS module 820 también irá conectada a la fuente de alimentación mediante cable de metal.

La Raspberry se conectará a la placa BAOS soldando dos cables con estaño. Uno de ellos irá desde el pin Rx de la placa hasta el pin 8 de la Raspberry, el puerto Tx. El otro cable empezará en el pin Tx de la placa y terminará en el pin 10 de la Raspberry, el puerto Rx. Dicho de otra forma, se realizará una conexión cruzada.

3.1.2 Programación de los dispositivos

Para la programación KNX sólo se puede usar la única herramienta que nos facilita KNX association para ello, **ETS5.**

Para comenzar, se debe tener alguna licencia de KNX para poder descargar este programa. Una vez descargado el programa ETS5, se abre y se procede a configurar el proyecto.

Para empezar, hay que crear un nuevo proyecto en este caso llamado Android.

Figura 22: Pantalla de inicio ETS5

Si no se tuviera un proyecto creado, únicamente se debería hacer clic en la cruz verde que se ve en la Figura 22 y poner en las opciones TP.

Una vez creado el proyecto, en este caso, se tiene que importar el dispositivo BAOS_KNX en él, ya que en el catálogo inicial del programa no está presente. Para realizar esto es muy sencillo, únicamente se tiene que ir al catálogo tal y como se muestra en la figura 23, darle a importar y seleccionar el archivo con extensión .knxprod en nuestro caso **KNX_BAOS_82x_87x.knxprod.**

Figura 23: Zona para insertar productos al catálogo

Una vez creado el proyecto e importado el producto KNX, se entra en él. Dentro del proyecto se va a las direcciones de grupos y se crean 3 direcciones de grupos para los distintos dispositivos que se van a conectar.

Después de esto se busca la placa KNX_BAOS en el catálogo y se le configura los parámetros según el tipo de dato que se reciba de las distintas direcciones de grupo. Dos de los parámetros creados son de tipo 2-bytes y flotantes que hacen referencia a los sensores de temperatura y deberemos asociarlos con sus respectivas direcciones de grupo.

Se ha añadido un parámetro más que haría referencia al sensor de consumo energético que es de 4-bytes flotante y lo asociamos a su dirección de grupo. Una vez creado todo esto hay que ir a la dirección de grupo principal hacer clic en él con el botón derecho y darle a programar y programación completa.

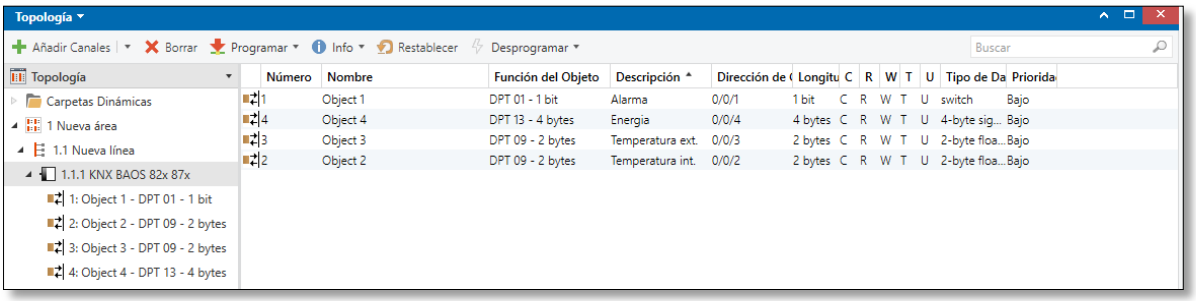


Figura 24: Parámetros configurados del proyecto

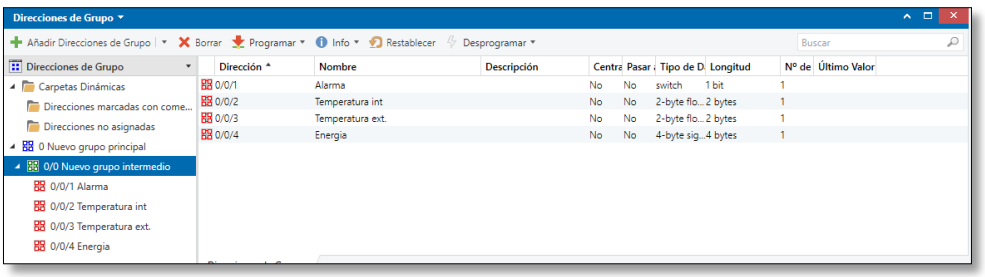


Figura 25: Direcciones de grupo del proyecto con sus correspondientes parámetros

En este caso como no se ha dispuesto de los sensores hasta el final del proyecto, por lo que se ha utilizado un dispositivo KNX llamado USB_KNX que conecta la placa BAOS con el ordenador a través de un USB como se aprecia en la figura 27, permitiendo hacer la simulación de los distintos sensores enviando datos desde el mismo ETS5 en la zona de diagnósticos. En él, se selecciona la dirección de grupo que se quiera y se escribe el dato que se quisiera enviar tal y como se aprecia en la figura 26.

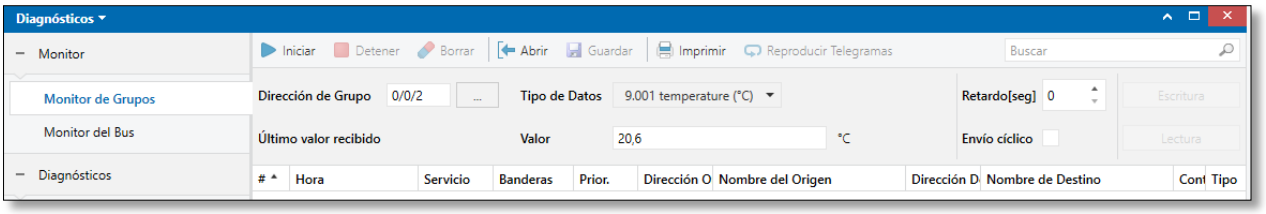


Figura 26: Datos que se envían desde KNX

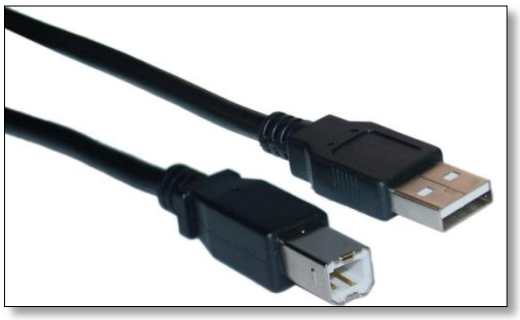


Figura 27: Cable USB usado para conectar el USB_KNX con el ordenador

3.2 Parte 2: Programación de la Raspberry Pi y programas para la recepción y envío de datos

3.2.1 Sistema Operativo Raspberry

La Raspberry Pi viene sin sistema operativo integrado por lo que si se quiere que actúe como un mini ordenador es necesario implementarle uno.

En este caso se ha instalado el sistema operativo Raspbian que es el oficial de Rapberry PI, y es una variación del Debian de distribución GNU/Linux. Para ello se ha ido a la página oficial de Raspberry Pi y se ha descargado la imagen del sistema operativo. Por otro lado, se ha cogido una tarjeta de memoria microSD vacía y con la ayuda del programa Win32DiskImager se ha montado la imagen en la tarjeta microSD.

Figura 28: Logo de Win32DiskImager

Por último, se tiene que insertar dicha tarjeta en la ranura que hay para ella en la Rapberry y al encenderla ya se tendrá puesto el sistema operativo.

Una vez instalado el sistema operativo, se configura la Raspberry para que pueda hacer lo que se requiere en este trabajo.

Para seguir programando la Raspberry, cabe destacar que se le ha añadido un ratón y un teclado en los puertos USB para facilitar su uso, se ha hecho que sea visible conectándolo a una pantalla vía cable HDMI y se ha conectado a la red internet poniéndole un cable RJ45 Ethernet conectado a un router.

Para empezar, se ha de cambiar la zona horaria de la Raspberry, ya que para este proyecto es esencial tener la fecha horaria de la instalación bien puesta. Para ello se ejecuta el siguiente comando en la terminal:

-sudo raspi-config

Tras ejecutarlo, saldrá una ventana con las distintas opciones de configuración de la Raspberry. Una vez dentro, se selecciona la opción Internationalisation Options, después se selecciona Change Timezone y se elige Europe y Madrid para establecer la fecha horaria.

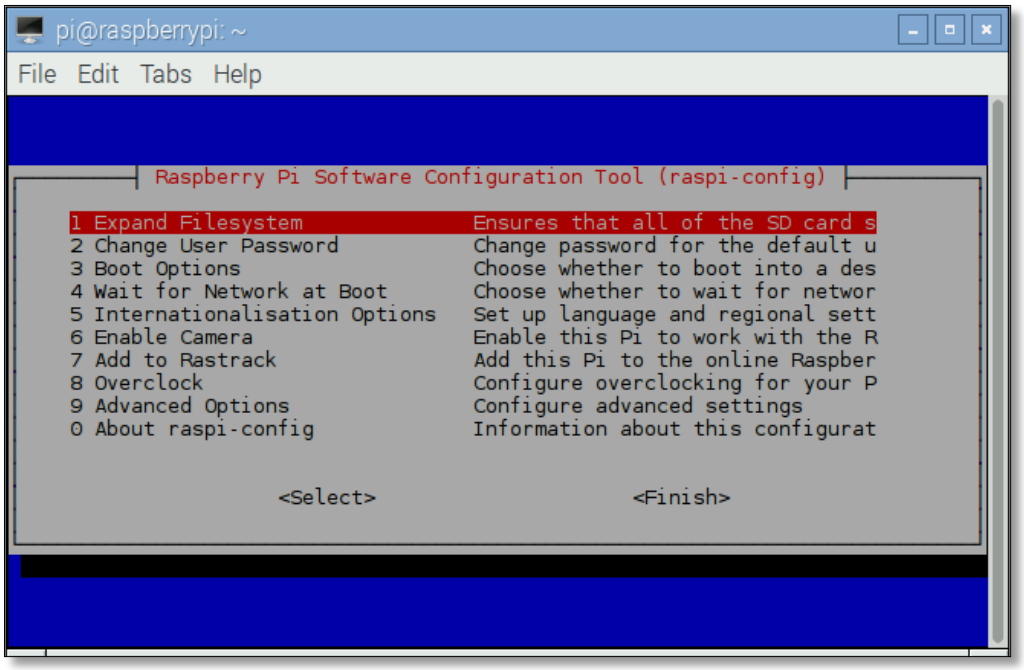


Figura 29: Zona de configuración de la Raspberry Pi

3.2.2 Instalar Java

El siguiente paso es la instalación de Java en la Raspberry. Para ello, únicamente se introducirá las siguientes líneas en la línea de comandos y se descargará la última versión de java disponible para un procesador ARM como el que lleva la Raspberry Pi:

-sudo apt-get upgrade

-sudo apt-get install Oracle-java8-jdk

3.2.3 Códigos Java

Como ya se ha explicado en apartados anteriores se ha creado un programa Java capaz de recibir los datos de los sensores y enviarlos a una base de datos en un dominio web. Los códigos del programa están en el anexo 2 de este proyecto.

El programa Java llamado KNX consta de tres clases dentro de un package llamado KNX:

- KNX.java: Se encarga de escuchar en el puerto serie y recibir las distintas tramas que le van llegando. Es la clase main encargada de llamar a las demás clases. Esta clase introduce los parámetros necesarios para recibir por los puertos serie y llama a la clase FrameReader.java para que analice las tramas recibidas y le devuelva ciertos valores, posteriormente llamará a un método dentro de esta clase que codificará los valores como objetos JSON y los enviará a la base de datos mediante un petición web Http.
- FrameReader.java: Esta clase hace la lectura de las tramas recibidas por la RaspBerry, espera a que le llegue un byte igual que el que indica el inicio de una trama según el módulo KNX_BAOS y va leyendo byte a byte toda la trama almacenando los bytes de interés en una variable del objeto Frame de la clase Frame.java. Una vez detecta que la trama está completa, esta clase le devuelve el objeto Frame a KNX.java. Las tramas que recibe el programa pueden ser de distintos tamaños depende del dato del sensor de que provenga. Esas tramas pueden contener datos de 2 o 4 bytes por lo que esta clase también convierte a double (formato de dato original) los distintos datos extraídos de las tramas.
- Frame.java: Esta clase se encarga de crear el objeto Frame en el que se almacena los datos de las tramas. Cabe destacar dentro de esta clase que contiene una variable byte Array donde se guarda toda la trama recibida.

Es reseñable decir que para la realización de estos códigos se ha hecho uso de librerías externas a Java que son:

- Json-simple-1.1.jar: Encargada de generar los objetos Json.
- RXTXcomm.jar: Se encarga de hacer la comunicación entre los puertos serie y Java.

La primera de estas librerías hay que descargarla de internet y hay que añadirla en el directorio donde Java tiene las librerías externas "usr/lib/jdk-8-oracle-arm32-vfp-hflt/jre/lib/ext" con el comando

-sudo mv librería lugardeorigen lugardestino

Para la segunda librería, en cambio, solamente se tiene que descargar desde el terminal con el siguiente comando:

-sudo apt-get install librxtx-java

El siguiente paso es compilar el programa Java, para ello es necesario meter todos los códigos Java en una carpeta creada en el escritorio a la que se ha llamado KNX. Para compilar los códigos es necesario acceder a dicha carpeta por la línea de comandos:

-cd /home/pi/Desktop/KNX

Una vez dentro, ejecutamos la siguiente línea:

-javac –cp ".:/usr/share/java/RXTXcomm.jar:/usr/lib/jni/librxtxSerial.so" KNX.java Frame.java FrameReader.java

Javac es el modo de compilar de Java desde la terminal y la opción –cp permite compilar los códigos java asociándolos a las librerías que se introducen en su interior.

Por último, antes de ejecutar el programa java se debe cambiar los permisos al puerto serie en la Raspberry, denominado como ttyAMA0, y modificar el archivo cmdline.txt situado en el directorio /boot eliminando toda referencia al puerto ttyAMA0, ya que por defecto la Raspberry es usado por el kernel del sistema y así se evita que de problemas a la hora de intentar acceder a él. Para cambiar los permisos se ejecuta la siguiente línea:

-sudo chmod 777 /dev/ttyAMA0

Y se quita toda la referencia al puerto editando el archivo antes mencionado con:

-sudo nano /boot/cmdline-txt

Dejando el archivo como aparece en la figura 30.

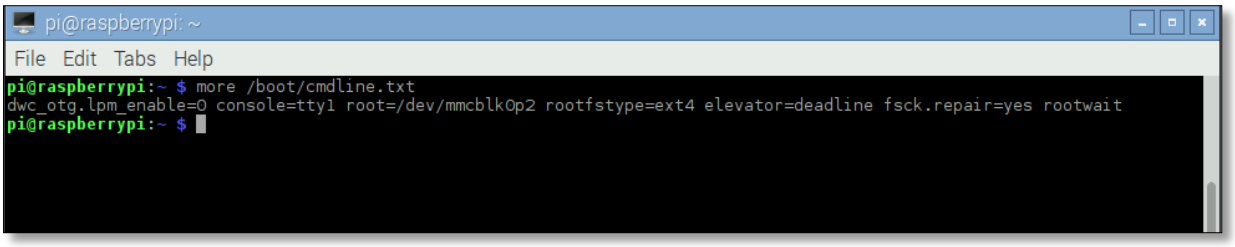


Figura 30: /boot/cmdline.txt modificado

Una vez se han hecho todos los pasos anteriores, hay que salir de la carpeta donde se encuentran los códigos Java y acceder al directorio donde se encuentra dicha carpeta. En ese mismo sitio es donde se debe ejecutar el programa java con el siguiente comando:

-java –Djava.library.path=/usr/lib/jni –Dgnu.io.rxtx.SerialPorts=/dev/ttyAMA0 KNX.KNX

Con la primera opción se indica donde están los archivos necesarios para ejecutar en este sistema operativo y con la segunda opción se abre el puerto serie ttyAMAO.

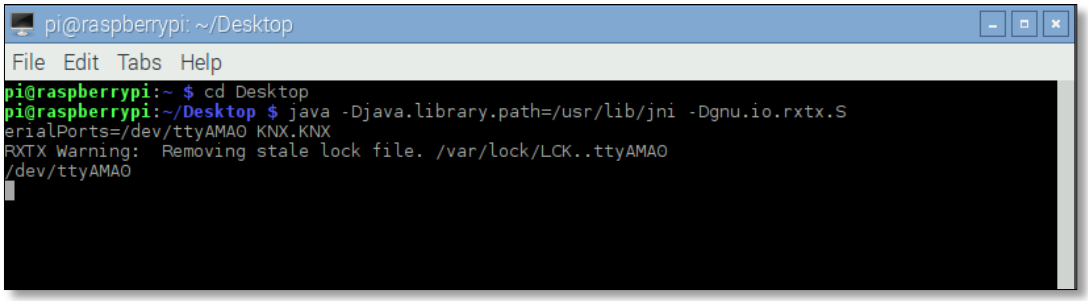


Figura 31: Programa Java preparado para escuchar las tramas

3.2.4 Envío de los datos a la base de datos

Como ya se ha comentado en apartados anteriores, el programa Java envía los datos recogidos por los sensores vía petición web. Esta petición web hace un POST a un PHP llamado insdatos.php.

```php
<?php


//conectar con la base de datos
$con = mysqli_connect("whattowear.es.mysql:3306","whattowear_es","CONTRASEÑA","whattowear_es");

if (!$con){

die('Could not connect: ' . mysqli_error());
}

//Recoger el POST, decodificar objeto Json y almacenar los valores en las variables
if(isset($_POST["dato"])){
    $json= $_POST["dato"];
    $obj = json_decode($json);
    $dato=$obj->{'dato'};
    $hora=$obj->{'hora'};
    $id=$obj->{'Id'};
    $fecha=$obj->{'fecha'};
    $timestamp=$obj->{'timestamp'};

//Insertar los valores en la tabla de datos
$sql="INSERT INTO datos_sensores (datos,Id,Fecha,Hora,TimeStamp) VALUES ('".$dato."','".$id."','".$fecha."','". $hora."','". $timestamp."')";
mysqli_query($con, $sql);
}

?>
```

Figura 32: Código PHP para insertar en la base de datos insdatos.php

Como se puede observar en la figura 30, el código PHP se conecta con la base de datos, recibe el POST, decodifica el objeto Json, guarda sus valores en variables y las inserta en la base de datos, en la tabla **datos_sensores.** Cada valor se inserta en su correspondiente campo.

La tabla datos_sensores de la base de datos contiene los siguientes campos:

- Datos(double): Aquí se guardarán los datos recogidos por los sensores.
- Id(int): Almacena de que dirección de grupo proviene cada dato.
- Fecha(date):Almacena las fechas en las que se ha tomado cada dato.
- Hora(time): Almacena las horas en las que se ha tomado cada dato.
- TimeStamp(timestamp): Almacena la fecha horaria en las que se ha tomado cada dato.

Browse | Structure | SQL | Search | Insert | Export | Import | Operations

#	Name	Type	Collation	Attributes	Null	Default	Extra	Action
1	datos	double			No	None		Change Drop More
2	Id	int(11)			No	None		Change Drop More
3	Fecha	date			No	None		Change Drop More
4	Hora	time			No	None		Change Drop More
5	TimeStamp	timestamp		on update CURRENT_TIMESTAMP	No	CURRENT_TIMESTAMP	ON UPDATE CURRENT_TIMESTAMP	Change Drop More

Figura 33: Tabla creada en la base de datos con sus correspondientes tipos de columnas

Para completar este apartado solo queda comentar el PHP que se encarga de recoger todos los datos de la base de datos y codificarlos como objetos Json cada vez que se haga una petición a este desde un dispositivo móvil. El PHP en cuestión se llama recogdatos.php y trata de: conectarse a la base de datos, seleccionar todos los elementos de la tabla **datos_sensores** y almacenarlos en un array de objetos Json.

```php
<?php
//Conectamos con la base de datos
$con = mysqli_connect("whattowear.es.mysql:3306","whattowear_es","CONTRASEÑA","whattowear_es");

if (!$con){
die('Could not connect: ' . mysqli_error());
}
//Seleccionamos todos los campos de la tabla
//ordenados según el timestamp de menor a mayor
$sql="SELECT datos,id,fecha,hora FROM datos_sensores ORDER BY TimeStamp ASC";

$datos=array();

$rs=mysqli_query($con,$sql);
//Guarda los objetos json en un array
while($row=mysqli_fetch_object($rs)){
    $datos[]=$row;

}
//imprime los datos
echo json_encode($datos);
?>
```

Figura 34: Código PHP para recoger los datos de la base de datos, recogdatos.php

3.3 Parte 3: Programación en Android

En esta última parte, se va a describir cómo se ha desarrollado la aplicación Android capaz de recibir los datos almacenados en la base de datos y cómo se han usado estos para poder ver de una manera sencilla la eficiencia energética.

Como en el apartado 3.1 todos los códigos generados se encuentran en la parte de anexos, en este caso en el anexo 3.

Para empezar hay que descargar el IDE Android Studio para poder programar en Java Android y poder crear los distintos layouts con sus correspondientes funciones. Una vez instalado el programa, se procede a crear el proyecto. Hay que seleccionar File y darle a new Project. Se escribe el nombre del proyecto, se selecciona el directorio en el que se quiera guardar y se avanza haciendo clic en siguiente. Se escoge Blank Activity, para tener el layout principal la más vacío posible. Una vez está el proyecto creado, se elimina todo el código de relleno que introduce Android Studio cada vez que se crea un nuevo proyecto.

3.3.1 Layouts

Este trabajo contiene tres layouts con los que se hara la aplicación completa:

- Activity_main: Es el layout inicial y el que se abre nada más iniciar la aplicación. Este layout constará de un Textview que contiene el nombre de la Aplicación y un Imageview donde se inserta una imagen alojada en la carpeta drawable del proyecto que ejercerá de botón para acceder a los siguientes layouts.
- Calendario: Este layout sirve como pasarela para acceder al layout de las gráficas. En este layout se filtra por días los datos que se quieran apreciar. Consta de 3 Textview donde se pide al usuario que seleccione las fechas que

desee. Además hay otros 2 Textview con un fondo diferente al fondo de pantalla donde una vez se seleccione una fecha se imprimirá en estos views. Por último, hay tres botones, dos que abren un diálogo con un calendario donde se podrá escoger las fechas y uno final que solo funcionará si se han escogido ambas fechas correctamente y llamará a la clase que carga el layout donde se encuentran las gráficas.

- Gráficas: Este último layout se encarga de sacar el resultado final por pantalla, es decir, las gráficas con sus datos correspondientes. En este layout hay 2 gráficas, una lineal y otra de barras verticales. La primera mostrará las temperaturas recogidas por los 2 sensores de temperatura según la fecha y hora en la que se han obtenido. La otra gráfica mostrará el consumo energético recogido por el sensor de consumo en cada día de los días seleccionados y también mostrará el consumo normalizado según la normalización de grados-día en dichos días. Para hacer esta normalización se usa un promediado de los datos de temperatura de la parte interior y exterior de la instalación. Con dichos promediados, se procede a hacer la siguiente fórmula que calcula los grados-día. Se basa en restarle 4 grados ºC al promediado de temperaturas internas, para obtener un temperatura de referencia. La temperatura del interior de la instalación suele ser la temperatura real que hay dentro de esta con un incremento entre 0 y 4 grados debido a diferentes factores como el calor humano de la gente que está dentro o la irradiación solar. A esta temperatura referencia se le restará el promediado de las temperaturas exteriores para ver la diferencia restante.

$$GradosDía = (PromedioTempInt - 4) - PromedioTempEext$$

Si el resultado es positivo, este valor es el factor por el que se divide el consumo. Si es negativo, el factor por el que se divide el consumo será de valor 1, no variando nada el resultado del consumo.

3.3.2 Clases Java

La aplicación consta de tres clases java que se deberán indicar dónde están situadas en el androidmanifest.xml tal y como se ve en el anexo3. Como se ha comentado en el apartado anterior, cada clase va relacionada con su layout. Esto se hace ya que cada clase extiende de Activity, que es lo que da acceso a los elementos del layout. En los métodos propios de Android como OnCreate cargaremos el layout correspondiente a la clase.

Las clases creadas para esta aplicación son las siguientes:

- MainActivity: Esta clase java es muy corta ya que sólo carga el contenido de layout Activity_main y consta de un método que carga la clase calendario una vez pulsamos el botón que aparece en el layout.

- Calendario: Esta clase se encarga de que el usuario seleccione un rango de fechas. El layout que se carga en esta clase es el llamado calendario. En dicho layout tenemos 2 botones que al presionarlos se llama a 2 métodos de esta clase que abrirán un calendario a modo de dialogo o popup, con un DatePicker que se encargará de recoger la fecha que seleccione el usuario y la mostrará por pantalla en los Texview con fondo blanco que tiene el layout. Dichas fechas se almacenarán en 2 variables diferentes; una que será la fecha inicial del rango y la otra la fecha final. Por ultimo hay un botón en la parte inferior del layout que al hacer clic en él se activará un método en esta clase que verificará si las fechas han sido escogidas, si son del mismo mes y si el día de fecha final es mayor al de la fecha inicial. Si cumplen todas las condiciones llamará a la clase gráficas y le pasará las variables que contienen la fecha de inicio y la fecha final del rango de fechas que se quiera graficar. Si no se cumplen las condiciones se lanzará otro dialogo explicando al usuario que debe seleccionar correctamente el rango.
- Gráficas: Esta clase se encarga de rellenar las gráficas del layout gráficas. Esta clase tiene más métodos que las anteriores clases, ya que debe hacer diferentes cosas. Para comenzar esta clase en el método OnCreate de Android que se inicia nada más activar el Activity el cual carga el layout de las gráficas que contiene las 2 gráficas a rellenar. En este método también se recoge los valores pasados de la clase anterior y se llama a un método que se encarga de hacer la petición web. En ese método crearemos un objeto cliente y le insertaremos los parámetros para realizar el POST. Una vez realizado el POST si este se ha efectuado de forma correcta llamaremos a un método de nombre organizarRespuestas al cual se le insertará un ArrayList de String creado con un método que se encarga de recibir los objetos Json y añadir sus valores de forma ordenada a la lista de Strings antes comentada. En el método de organizarRespuestas se realiza el filtrado por fechas ya que analiza las fechas de la lista de Strings y solo se cogen los valores que entran en el rango. Una vez hecho este filtrado se procede a ordenar los datos de la lista de respuestas mirando su Id, es decir su dirección de grupo, en este caso si el id es 2 o 3 se añade a una lista de temperaturas, sus correspondientes fechas se insertan en otra lista en la que irán las fechas de los datos y sus ids que irán en otra lista. Cabe decir que ambas listas están en concordancia ya que cada vez que se inserta un dato se inserta la fecha y el id también. Si el id es 4 los datos se añadirán a la lista de consumo y sus correspondientes fechas a la lista de las fechas del consumo. Cuando se termina de realizar estas listas se llama a dos métodos Tempgraf y Consgraf que se van a encargar de crear las gráficas e introducir los datos en ellas. Al método Tempgraf se le introducirán las listas con los datos sus fechas y sus ids. En este método se creará 2 listas nuevas diferenciando si los datos de las temperaturas vienen del sensor interior o de la sonda exterior. Con esas listas y la librería mpandroidchart se irá creando las entradas para la gráfica y se modificará esta para sacar una gráfica de 2 líneas que mostrará los datos de la temperatura interior y los de la temperatura exterior ordenadas de forma temporal. El otro método encargado de rellenar la gráfica que queda, la de

barras, es Consgraf, en el que se añaden todas las listas que se crean en organizarRespuestas. Hay que añadirlas todas ya que son necesarias para hacer la gráfica de consumo normalizado según los grados-día. Para empezar en este método como en el método anterior creamos las listas con los datos de la temperatura interior y con los datos de la temperatura exterior, a la vez que creamos otras 2 listas con las fechas de estos datos. Una vez creadas estas listas se procede a calcular los grados-día de cada día para ello se sumarán todos los valores de los datos de temperatura del mismo día y se dividirán por la cantidad de datos que hay en esos días. Cuando se hacen estos promediados se irán almacenando en unas listas que almacenarán los promedios diarios de las temperaturas interiores y los de las temperaturas exteriores. El siguiente paso es crear la entradas para la barra que se encarga de visualizar el consumo diario para ello cogeremos el último valor del consumo del día y le restaremos el último valor del día anterior, ya que el dato del consumo es un dato acumulativo de forma que lo consumido en ese instante se le suma lo consumido anteriormente, de esta forma se consigue obtener el dato consumido en un día. Estos datos se almacenarán en otra lista encargada de ellos. Por último este método se encarga de crear las entradas para la barra encargada de visualizar el consumo normalizado por los grados-día. Para ello se coge las listas de los promedios y se calculan los grados-día según se hace en la fórmula indicada en el apartado 3.3.1 en el layout gráficas. Calculados los grados-día solo queda dividir el consumo del día por su correspondiente grado-día si el grado-día es positivo y por 1 si el valor del grado-día es negativo. Los valores resultantes se añadirán a la lista de entradas de la barra comentada anteriormente.

3.3.3 Librerías añadidas

Estas clases dependen de librerías externas por lo que se deberán añadir en el build.gradle de la aplicación. Las librerías que se han añadido son mpandroidchartlibrary-2-1-6.jar y como en el caso de los códigos java, la librería Json-simple-1.1.jar.

- mpandroidchartlibrary-2-1-6.jar: Esta librería se encarga de ofrecer un elemento capaz de hacer ver gráficas en el layout a la vez que proporciona diferentes funciones capaces de poder manipular y crear dichas gráficas. Esta librería permite crear diferentes tipos de gráficas; graficas de barras, gráficas circulares, gráficas lineales, etc. En este proyecto se hace uso de la gráfica de líneas y la gráfica de barras que son las que mejor hacen ver el resultado de este trabajo.
- Json-simple-1.1.jar: En este caso esta librería se usa para decodificar los objetos Json y almacenarlos en variables de tipo String.

Para insertar librerías en este proyecto y crear las dependencias necesarias con el fin de que la aplicación las reconozca, se debe copiar los archivos .jar en la carpeta libs de la aplicación.

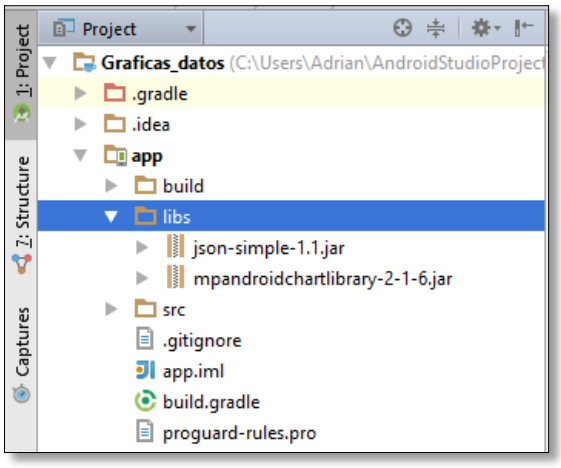


Figura 35: Carpeta de librerías en Android Studio

Una vez copiados los archivos hay que ir a file y seleccionar Project Structure. Dentro de Project Structure hay que seleccionar la app e ir al apartado de dependencias. En este apartado se añaden las dependencias haciendo clic en la cruz verde que aparece en la parte derecha de la figura 37, se escoge la opción file dependency y añadir ambas librerías.

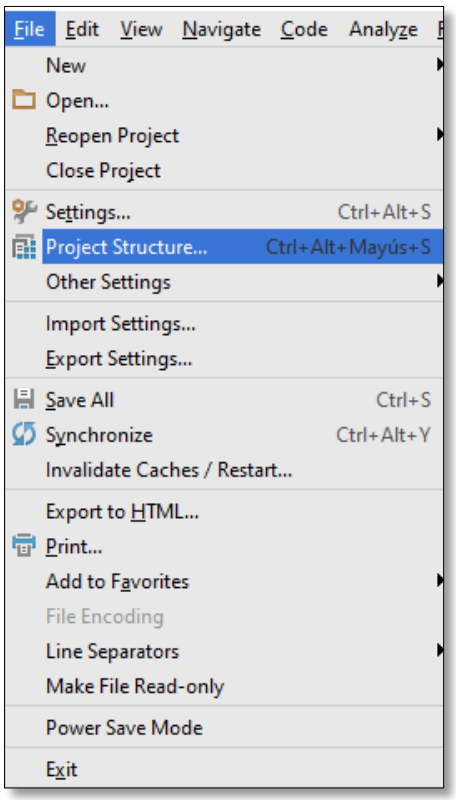


Figura 36: Localización de Project Structure en Android Studio

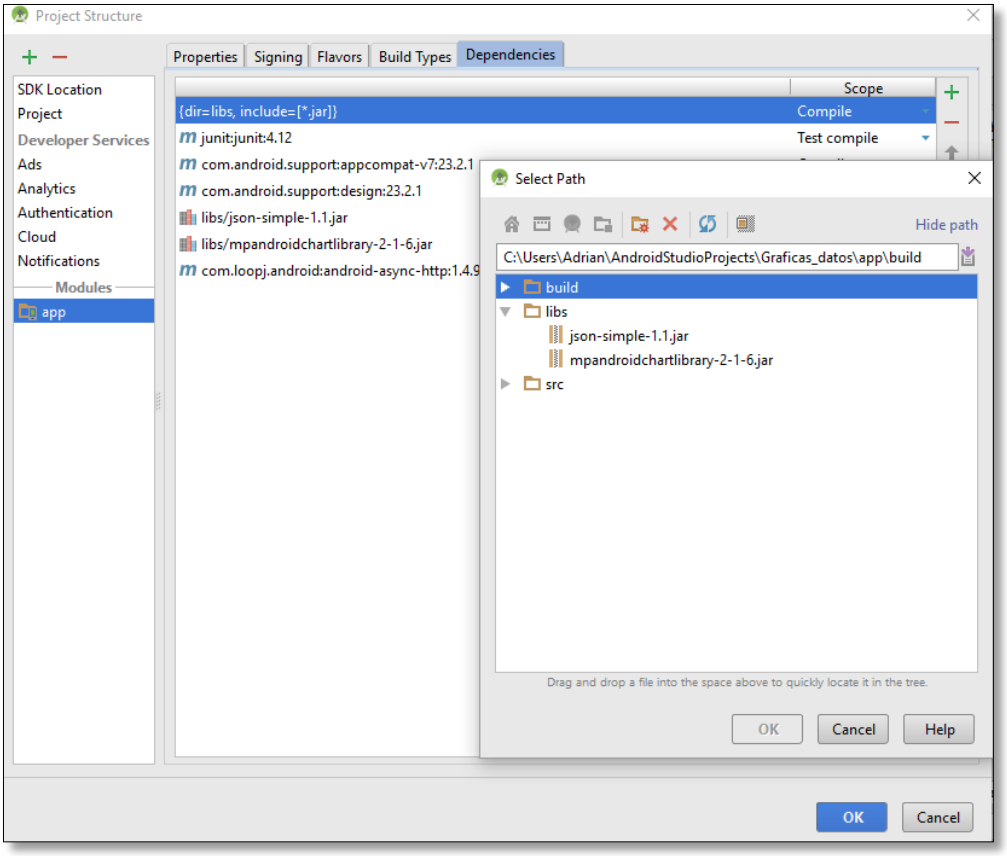


Figura 37: Añadir librerías externas en Android Studio

Por último, queda comentar que para hacer las peticiones web se debe añadir una última librería llamada com.loopj.android: android-async-http:1.4.9. Esta librería está dentro de Android Studio y para añadirla sólo hay que realizar los mismos pasos antes comentados y en vez de escoger la opción de file dependency, escoger la opción de library dependency y buscar la librería.

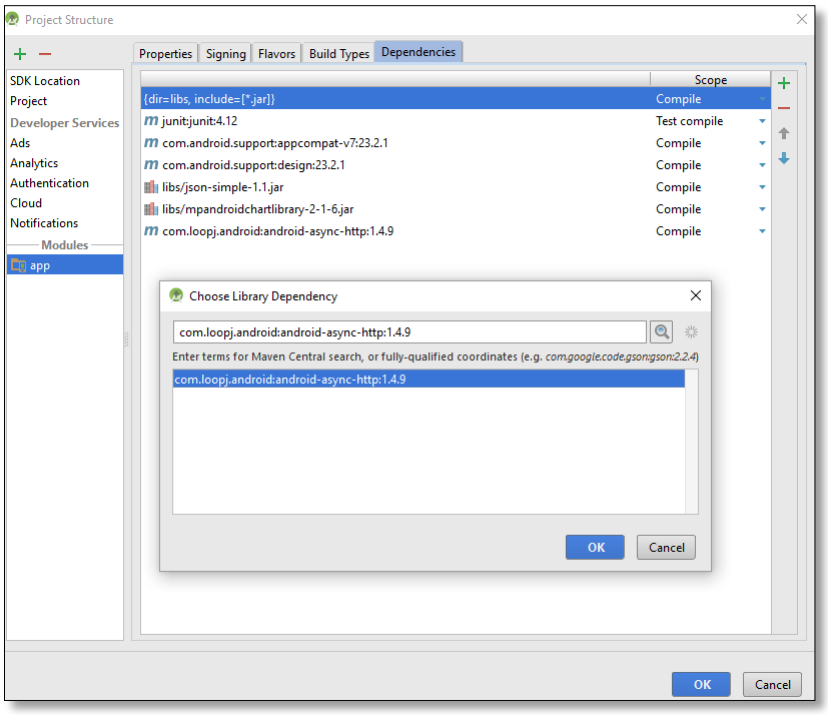


Figura 38: Añadir librería interna Android Studio

Parte 4: Resultados

En la parte de resultados se va a proceder a visualizar y comentar los resultados tanto finales como intermedios del trabajo. Para empezar, en este apartado se expondrán los resultados que se obtienen al recibir un dato de cada sensor en el programa Java y cómo los va almacenando en la base de datos. Este apartado seguirá mostrando el resultado de la aplicación de sus primeros 2 layouts, Activity_main y Calendario. Finalizará mostrando el resultado final con las gráficas que se aprecian en el layout Gráficas.

4.1 Resultados del programa Java y base de datos

Como se ha comentado anteriormente, esta sección mostrará los datos obtenidos al recibir los datos de los distintos sensores y cómo el programa los almacenará en la base de datos. Comentar que para poner a escuchar el programa Java hay que seguir los pasos indicados en el apartado 3.2.3.

4.1.1 Datos del sensor interior

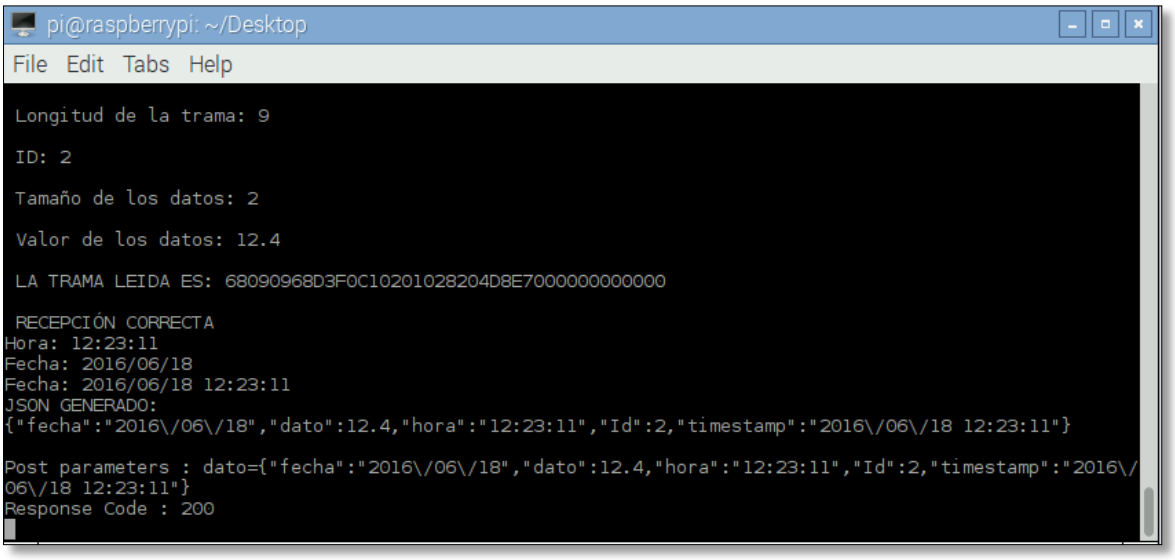


Figura 39: Dato recibido de sensor de temperatura interior

Como se puede apreciar en la figura 39 hemos recibido una trama de tamaño 9 bytes y con Id igual a 2, que es el correspondiente al sensor de temperatura colocado en el interior de la instalación. También se puede apreciar el valor de los datos y cómo es la trama recibida.

Después de analizar la trama, el programa Java procede a crear un objeto Json con los datos de la trama, la fecha y la hora en la que se han recibido. Además, también se puede observar el objeto Json con el que se hace la petición web POST y si éste se ha enviado correctamente o no.

4.1.2 Datos del sensor exterior

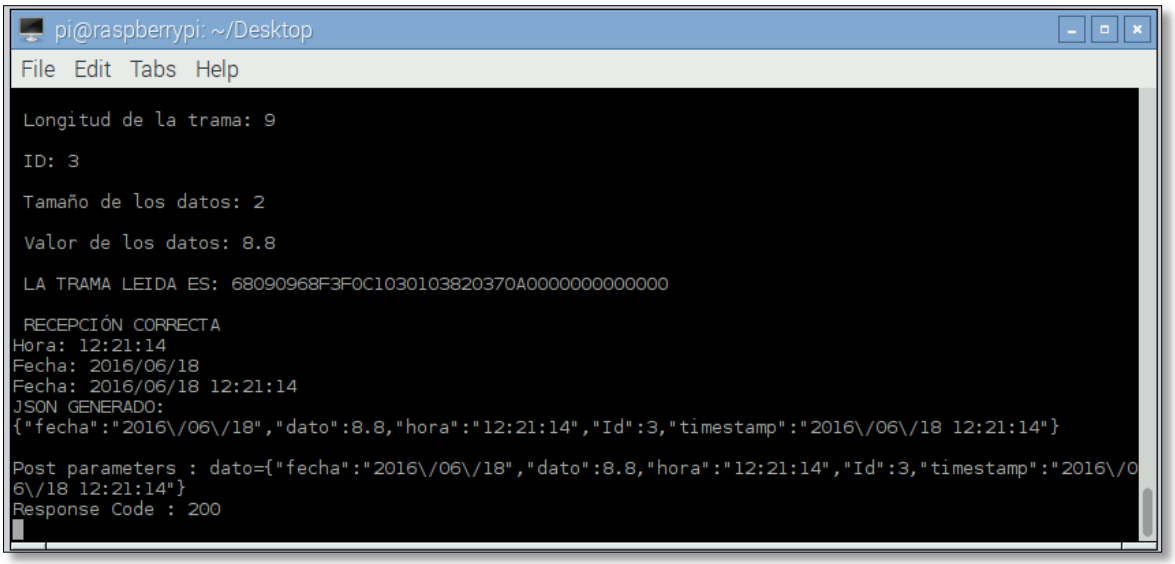


Figura 40: Dato de la sonda que recoge la temperatura exterior

Este resultado no se distingue mucho del apartado previo salvo en que el Id de la trama en esta ocasión es el tres, el cual hace referencia a la sonda de temperatura colocada en el exterior de la instalación. Otra de las diferencias es el valor del dato que en este caso es diferente al anterior. Por lo demás, es igual que lo descrito en el apartado precedente, se recoge la hora y fecha de la recepción de datos, se construye el objeto Json y se ve si ha sido enviado correctamente.

4.1.3 Datos del sensor de consumo

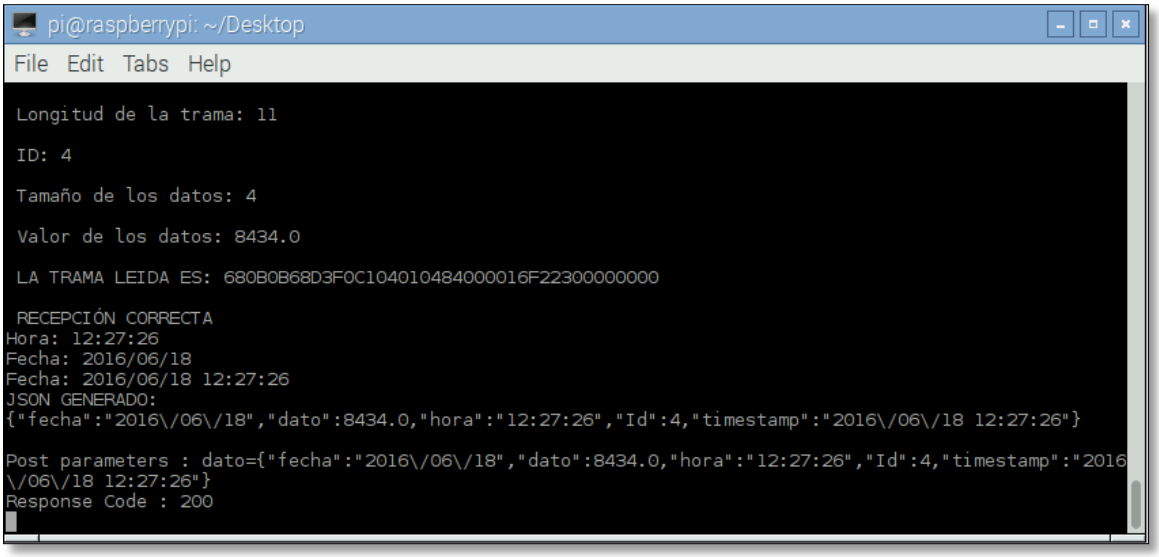


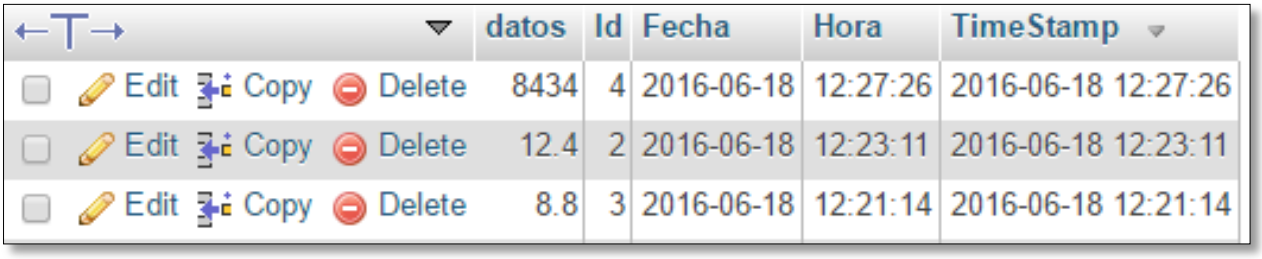
Figura 41: Dato obtenido del sensor de consumo

Este resultado varía un poco más de los apartados anteriores ya que el tamaño de los datos obtenidos es mayor que en los apartados anteriores. Se puede observar que el tamaño de esta trama es mayor que en los anteriores apartados, 11 bytes, ya que el tamaño de los datos es de 4 bytes. También se puede apreciar que el Id en esta trama es diferente a las anteriores. En este caso el Id es 4, que hace referencia al sensor de consumo de la instalación. Al ser el tamaño de datos de 4 bytes el rango de valores que abarca este tipo de dato, es mayor que el de los anteriores casos.

El resto del resultado es el mismo que en los apartados anteriores, se recoge la hora y fecha de la recepción del dato, se crea el objeto Json y se ve si ha sido enviado correctamente.

4.1.4 Base de datos

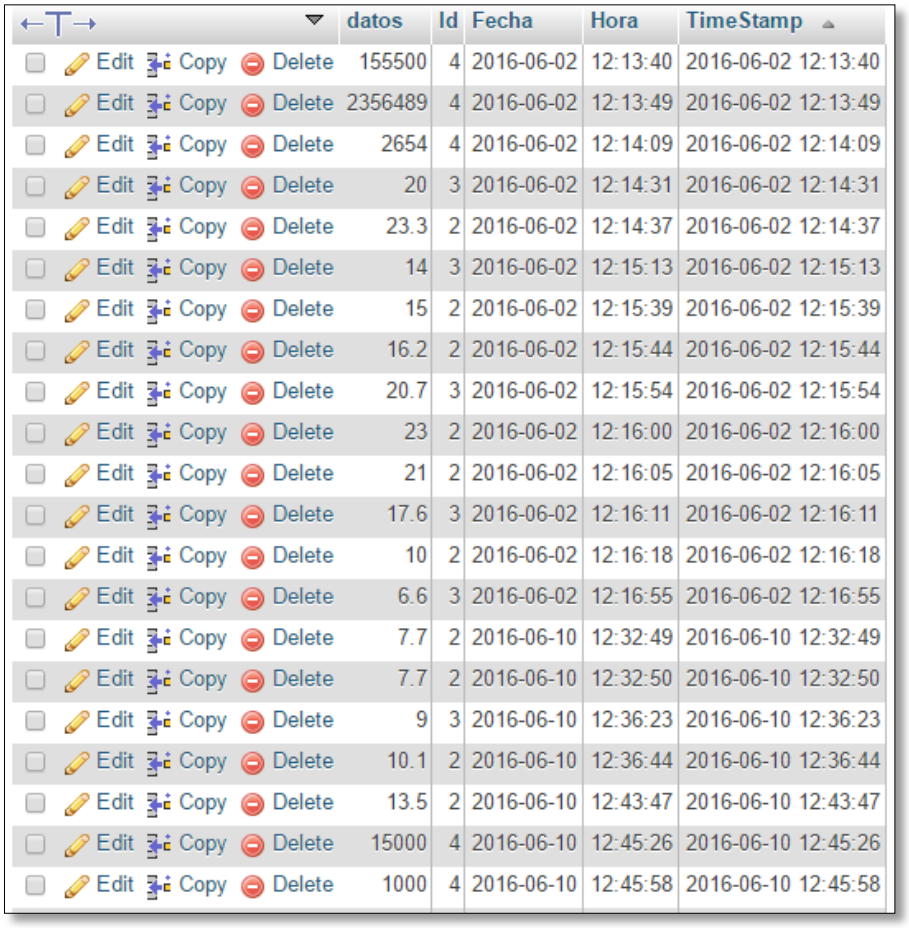
En este apartado se aprecia cómo los datos recibidos se van almacenando en la base de datos mediante los PHP a los que hace la petición el programa Java.



	datos	Id	Fecha	Hora	TimeStamp
Edit Copy Delete	8434	4	2016-06-18	12:27:26	2016-06-18 12:27:26
Edit Copy Delete	12.4	2	2016-06-18	12:23:11	2016-06-18 12:23:11
Edit Copy Delete	8.8	3	2016-06-18	12:21:14	2016-06-18 12:21:14

Figura 42: Datos insertados en la base de datos

Como se puede apreciar en la figura 42, en la base de datos se van insertando el valor de la trama, su Id, la fecha y la hora en las que han sido tomadas cada una más un timestamp que luego nos servirá para recoger las fechas ordenadamente desde la primera hasta la última.



	datos	Id	Fecha	Hora	TimeStamp
Edit Copy Delete	155500	4	2016-06-02	12:13:40	2016-06-02 12:13:40
Edit Copy Delete	2356489	4	2016-06-02	12:13:49	2016-06-02 12:13:49
Edit Copy Delete	2654	4	2016-06-02	12:14:09	2016-06-02 12:14:09
Edit Copy Delete	20	3	2016-06-02	12:14:31	2016-06-02 12:14:31
Edit Copy Delete	23.3	2	2016-06-02	12:14:37	2016-06-02 12:14:37
Edit Copy Delete	14	3	2016-06-02	12:15:13	2016-06-02 12:15:13
Edit Copy Delete	15	2	2016-06-02	12:15:39	2016-06-02 12:15:39
Edit Copy Delete	16.2	2	2016-06-02	12:15:44	2016-06-02 12:15:44
Edit Copy Delete	20.7	3	2016-06-02	12:15:54	2016-06-02 12:15:54
Edit Copy Delete	23	2	2016-06-02	12:16:00	2016-06-02 12:16:00
Edit Copy Delete	21	2	2016-06-02	12:16:05	2016-06-02 12:16:05
Edit Copy Delete	17.6	3	2016-06-02	12:16:11	2016-06-02 12:16:11
Edit Copy Delete	10	2	2016-06-02	12:16:18	2016-06-02 12:16:18
Edit Copy Delete	6.6	3	2016-06-02	12:16:55	2016-06-02 12:16:55
Edit Copy Delete	7.7	2	2016-06-10	12:32:49	2016-06-10 12:32:49
Edit Copy Delete	7.7	2	2016-06-10	12:32:50	2016-06-10 12:32:50
Edit Copy Delete	9	3	2016-06-10	12:36:23	2016-06-10 12:36:23
Edit Copy Delete	10.1	2	2016-06-10	12:36:44	2016-06-10 12:36:44
Edit Copy Delete	13.5	2	2016-06-10	12:43:47	2016-06-10 12:43:47
Edit Copy Delete	15000	4	2016-06-10	12:45:26	2016-06-10 12:45:26
Edit Copy Delete	1000	4	2016-06-10	12:45:58	2016-06-10 12:45:58

Figura 43: Datos de varios días en la base de datos

4.2 Resultado de los Layouts Activity_main y Calendario

En esta parte se va a mostrar el resultado obtenido de la creación de los layouts y qué aspecto tendrá en un dispositivo móvil. Los explicaremos según el orden lógico que sigue la aplicación, es decir, primero se explicará el layout Activity_main y luego se explicará el layout calendario.

4.2.1 Layout Activity_main

Figura 44: Pre visualizado de layout Activity_main

Lo que se puede apreciar en la figura 44 es el pre visualizado que ofrece Android Studio de los layout que se van creando en el proyecto en este caso, es el Activity_main. Como se puede ver en la siguiente figura, el resultado final en un dispositivo móvil no dista mucho de la pre visualización. En ambas figuras la pantalla tiene un fondo color salmón suave pudiendo distinguir claramente los elementos que hay en ella. El primer elemento es el título de la aplicación que está escrito en negrita y con color rojo para resaltar más del fondo de pantalla, cabe decir que este elemento está colocado en la parte superior-central de la pantalla y su alineamiento de texto es céntrico. El siguiente elemento que aparece es una imagen de un termómetro en color rojo para seguir con el estilo de la aplicación y colocado en la parte central de la pantalla. Dicha imagen actuará como un botón y al ser pulsada se lanzará el siguiente layout o pantalla de la aplicación.

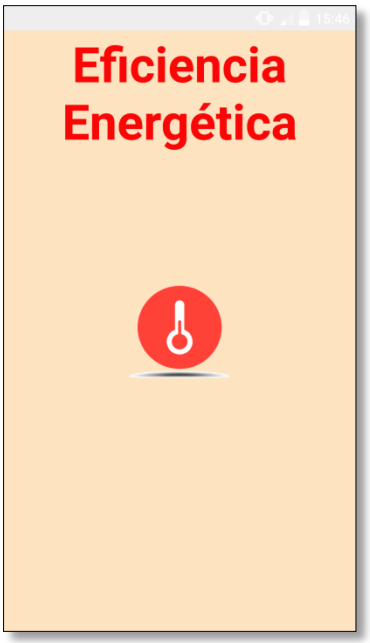


Figura 45: Resultado final del layout Activity_main

4.2.3 Calendario

Figura 46: Pre visualizado del layout calendario

En la figura 46, como en el apartado anterior, se puede apreciar el pre visualizado que otorga Android Studio del layout Calendario. Como en el caso anterior este tampoco dista mucho del resultado final en un dispositivo móvil, solo que en este layout se irá modificando a la vez que el usuario accione los distintos elementos. Como podemos ver en la figura 46, en este layout tenemos 8 elementos diferenciados. El primer elemento de este layout es un texto que pide al usuario que escoja el rango de fechas y está situado en la parte superior-central de la pantalla y el texto tiene un alineamiento céntrico. Los siguientes tres elementos van en conjunto, lo mismo que los tres

elementos posteriores a estos. Estos conjuntos sirven para escoger las fechas y varía uno del otro en el primer elemento del conjunto, que es un texto, en el primer conjunto pone **Desde:** que indica que se escoja la primera fecha y en el segundo caso pone **Hasta**: que indica que escoja hasta que fecha quiere hacer el rango. El siguiente elemento de ambos conjuntos es una zona con fondo blanco diferente al fondo de pantalla que simula un cuadro de texto que una vez seleccionadas las fechas se rellenarán con dicha fecha. El último elemento de estos conjuntos es un botón que una vez se presiona lanzará un diálogo o popup. El diálogo que aparece es un calendario con un Datepicker que se encarga de saber qué fecha es seleccionada. Una vez que se han seleccionado las fechas, éstas se visualizarán en el campo de texto antes comentado según qué botón se haya presionado. El elemento final de este layout es un último botón que si las fechas han sido escogidas correctamente, es decir, que ambas fechas se hayan seleccionado, que sean del mismo mes y que el día de la primera fecha sea menor que el de la última, llevará al layout de las gráficas. Si alguna de estas condiciones no se cumplen, el botón lanzará diálogos de texto indicando al usuario qué debe hacer para que funcione correctamente.

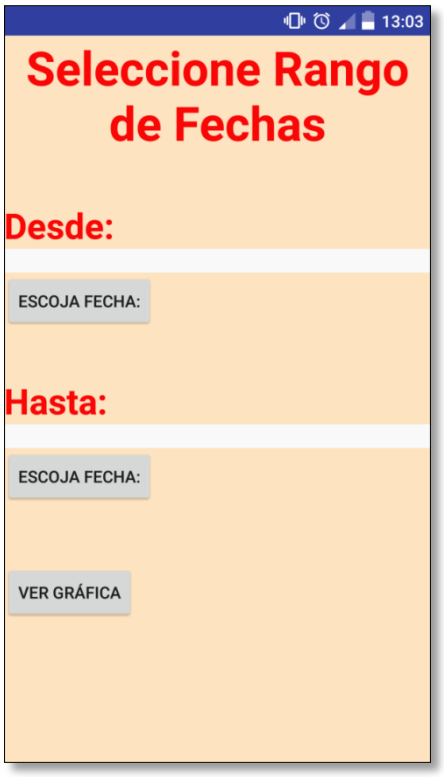


Figura 47: Resultado final layout calendarío

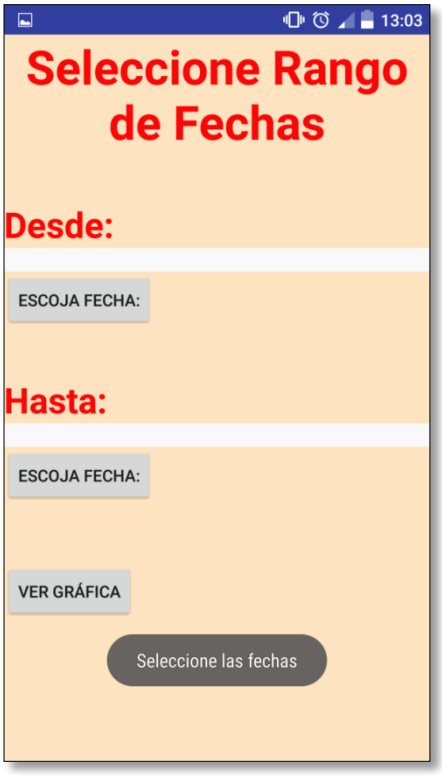


Figura 48: Función de botón ver gráfica sin fechas seleccionadas

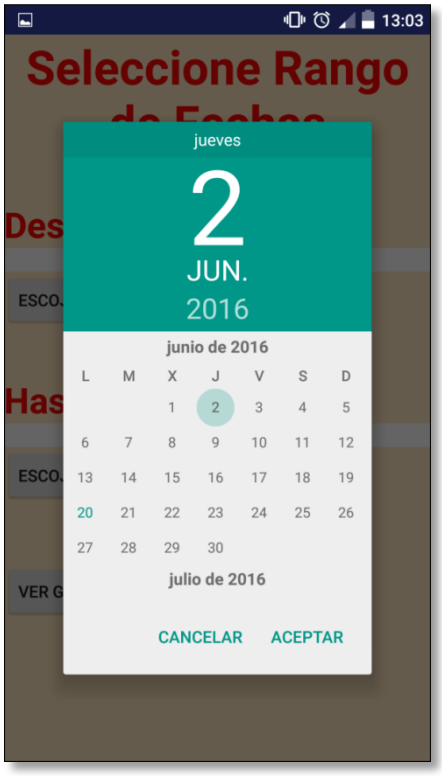


Figura 49: Dialogo con el calendario que se abre al presionar el botón escoja fecha

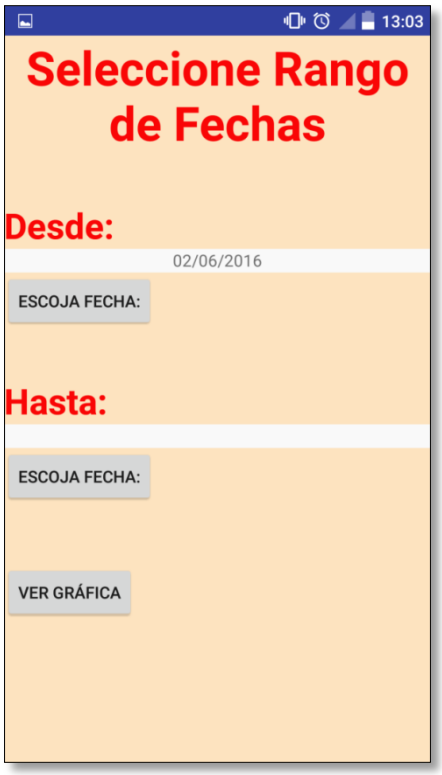


Figura 50: Layout calendario con una fecha escogida

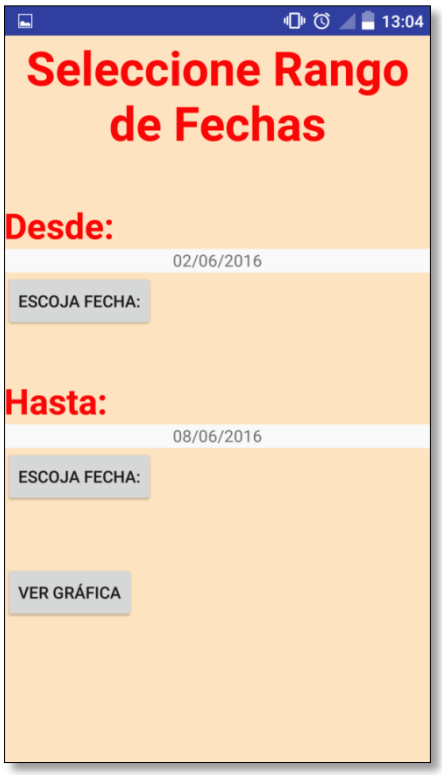


Figura 51: Layout calendario con ambas fechas escogidas correctamente

Figura 52: Dialogo de calendario escoger fecha de un mes diferente

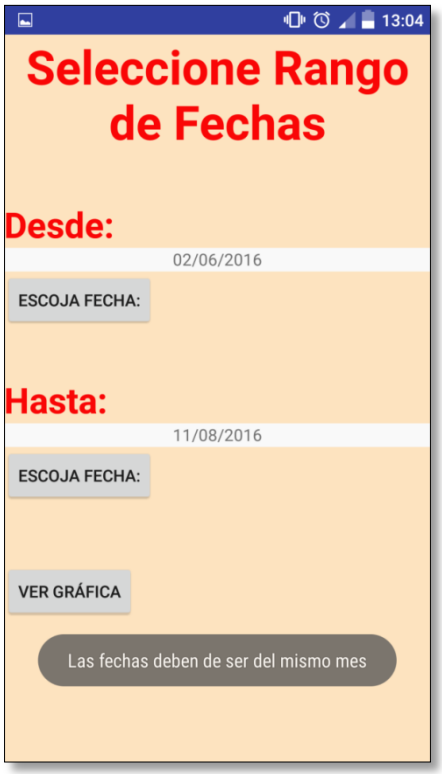


Figura 53: Función del botón ver gráfica con las fechas de distintos meses

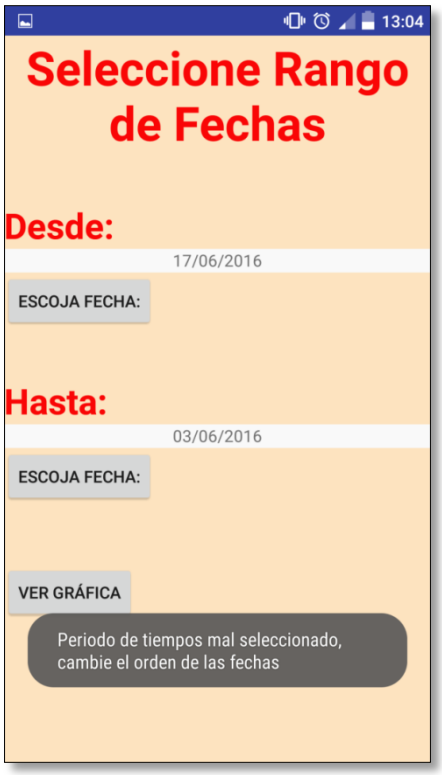


Figura 54: Función del botón ver gráfica con las fechas escogidas del mismo mes pero con las fechas mal

4.3 Resultado del layout Gráficas

Este layout es el último layout de la aplicación y se encarga de mostrar las gráficas obtenidas con los datos recibidos y donde se pueden observar los resultados de la aplicación.

Figura 55: Pre visualizado del layout gráficas

Como se puede ver en la figura 55, en el pre visualizado que otorga Android Studio, este layout está dividido en 2 zonas. Ambas zonas son las que hacen referencia a las gráficas pero no se puede obtener un visualizado de estas ya que cada vez que se accede a este layout a través del layout calendario los datos que se visualizan son diferentes, y con dichos datos se van creando ambas gráficas. En este apartado, a diferencia del apartado 4.2, el resultado final de la aplicación varía mucho de la pre visualización de Android Studio, como se podrá apreciar más adelante.

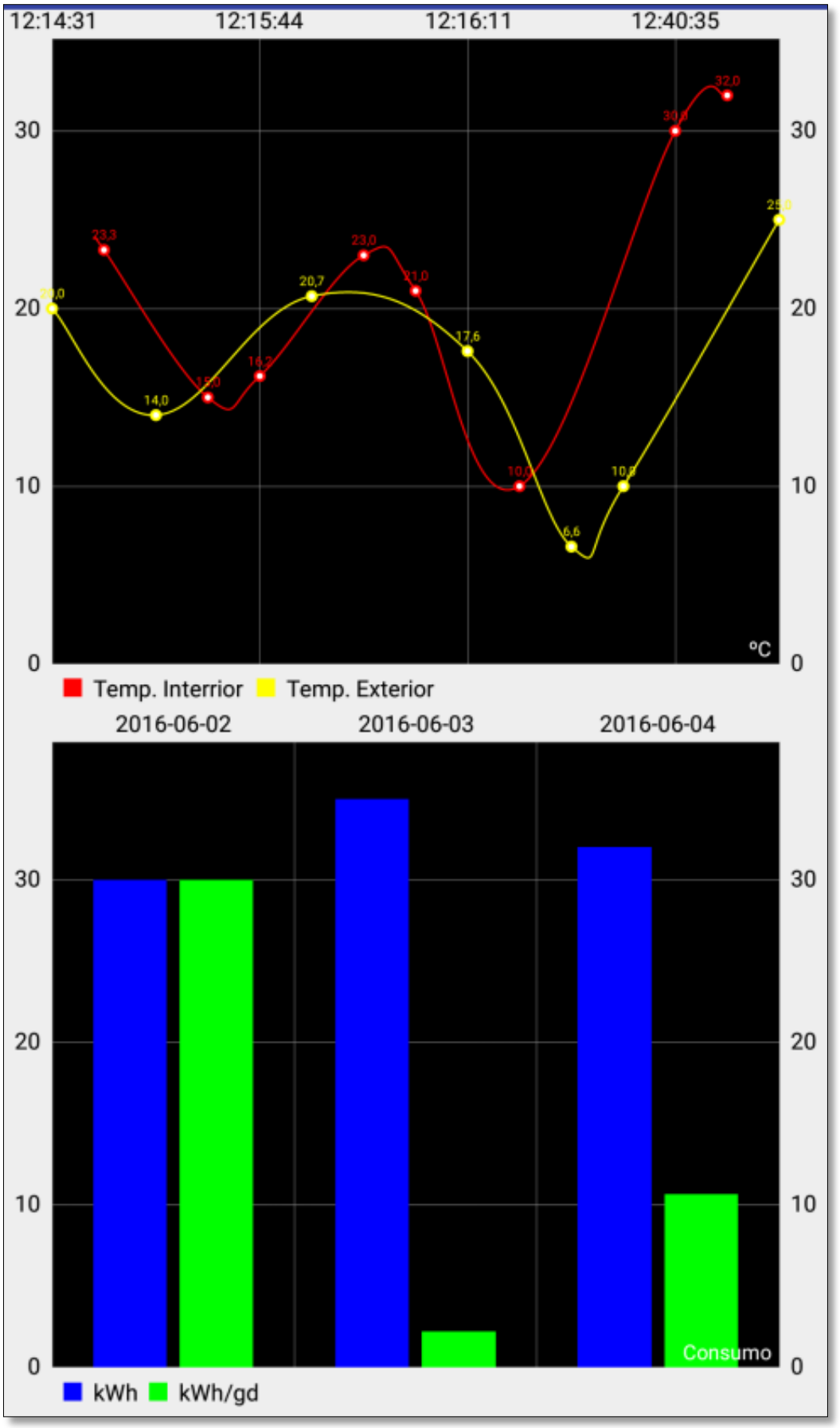


Figura 56: Resultado final del layout gráficas

El resultado final de este trabajo se puede apreciar en la figura 56.

Se observan 2 gráficas perfectamente divididas una de la otra. La primera de las gráficas muestra los datos de temperatura interior, línea roja, y los datos de temperatura exterior, línea amarilla, de forma que se puede ver las variaciones de temperatura de forma sencilla. En dicha gráfica se muestran los valores de los datos y la hora en que han sido obtenidos. Estos valores son ordenados de modo que los de la fecha anterior irán situados a la izquierda de la gráfica y los de la fecha posterior en el lado derecho. Los cambios de día se pueden observar cuando la hora pasa de 00:00 a la 01:00. Si la gráfica contiene varios datos se podrá hacer zoom en ella y evaluar las zonas que se deseen.

La segunda gráfica muestra los datos del consumo de los días del rango seleccionado a la vez que se muestra el consumo normalizado por grados-día de dichos consumo. La barra de color azul es el consumo del día en kWh y la barra verde es el consumo normalizado por grados-día en kWh/gd. Cabe destacar que en la figura 56 la primera barra del consumo normalizado por grados-día tiene un valor de grado-día inferior a 0 y por tanto dicha barra es igual a la del consumo del día. Las siguientes barras del consumo normalizado por los grados-día tienen un valor mayor que 0 y por tanto dichas barras variarán de sus correspondientes gráficas de consumo.

Cabe destacar que el resultado que se visualiza en la figura 56 es un resultado con datos no reales y transmitidos por el dispositivo KNX_USB comentado en apartados anteriores para dar la validación y ver que la aplicación creada funciona de forma correcta una vez se lleve a cabo de forma real. No se puede tener datos reales por que los dispositivos KNX no se han podido usar de forma continua para sacar un resultado real teniendo en funcionamiento constante todos los dispositivos varios días consecutivos.

Parte 5: Conclusiones

Una vez finalizado el trabajo es momento de realizar las conclusiones pertinentes, evaluar su utilidad en el mundo laboral, analizar las herramientas utilizadas y valorar el resultado final.

Para empezar se enunciará por qué para la realización de este proyecto se han usado los materiales comentados en la parte 2 de trabajo:

- KNX: Actualmente es la compañía líder en Europa, zona donde se pretende llevar a cabo este trabajo y actualmente engloba el 80% de los dispositivos en ella. Esto hace que sea más interesante trabajar con este protocolo de comunicación de red ya que la mayoría de instalaciones domóticas en Europa requerirán de sus dispositivos.
- Raspberry Pi: Se ha optado por trabajar con la Raspberry Pi en este trabajo debido a que ésta puede trabajar como un mini ordenador a la vez que nos permite con sus puertos serie o UART conectarnos al bus KNX mediante la placa KNX_BAOS module 820. También se ha optado por su uso debido a que su sistema operativo oficial está preparado para trabajar con el lenguaje de programación Java con las que podemos hacer el funcionamiento de este trabajo.
- Java: Java es el lenguaje líder de programación orientada a objetos y por tanto tiene más facilidades y más herramientas (librerías) con las que dotan al usuario que lo utilice de más opciones a la hora de programar. En el caso de este trabajo se necesitaba una librería para poder utilizar los puertos serie de la Raspberry Pi (RXTXcomm.jar) muy implementada en códigos Java y de donde más información se podía obtener.
- Android: En este trabajo se ha optado por crear una aplicación en Android debido a que la gran mayoría de los dispositivos móviles tienen este sistema operativo y dado que el lenguaje de programación en Android es una variación de Java adaptada para Android.
- Android Studio: Se ha trabajado con Android Studio debido a que en asignaturas de la carrera se ha utilizado esta herramienta de programación para programar en Android y a priori el conocimiento de ésta era mayor que de las demás herramientas de programación en Android.

De los resultados obtenidos, podemos destacar que es una aplicación que de forma muy sencilla te permite ver la eficiencia energética de la instalación, ya que saca los resultados obtenidos a modo de gráfica.

Una de las gráficas muestra las variaciones de la temperatura tanto del interior como del exterior, con las que se pueden apreciar mejor y dar más veracidad al gráfico del consumo normalizado por grados-día. Ya que podremos observar cuanta temperatura teníamos en el interior de la instalación y cuanto en el exterior y ver si distan mucho una de la otra.

Por otro lado tenemos la gráfica de barras que muestra la variación del consumo energético a lo largo de los días y el consumo normalizado por grados-día. La normalización por grados-día permite tener un dato más fiable del consumo energético ya que este tiene en cuenta las temperaturas en la zona. Se podría tener un consumo muy elevado, pero este podría ser de un día de invierno en el que haga bastante frio por lo que ese dato sería un consumo normal. Dicho esto, las barras pertenecientes a los datos de consumo por grados-día serán todas parecidas de valor, por lo que cuando haya alguna barra que destaque más que las otras podremos observar alguna variación en el consumo, es decir, si se ha consumido más de lo necesario o menos. El día que tengamos un grado-día alto significará que se está usando energía de más y la barra será muy pequeña; En cambio, cuando el grado-día sea bajo, significará que no hemos consumido mucho y por tanto la barra de grados día será más pronunciada.

El resultado de este trabajo pretende ser el mismo que cuando una empresa instala un sistema de domótica hace para que se pueda apreciar si el sistema domótico correspondiente funciona correctamente, sin entrar en temas técnicos.

Cabe decir que el resultado obtenido de esta aplicación actualmente se hace en auditorías de eficiencia, por lo que esta aplicación ahorraría el gasto que supone contratar dicha auditoría y las horas de trabajo que requiere.

Otro detalle a destacar es que esta aplicación no tiene una temperatura de referencia estática y fija como las que suelen utilizar en este tipo de auditorías. En esta aplicación la temperatura referencia varía según el promediado de las temperaturas del interior, por lo que dota a este trabajo un resultado más real que en dichas auditorías.

En modo de punto final, comentar que este proyecto sólo mide la eficiencia energética térmica de una instalación en una única zona. Por lo que si se quisiera proceder a desarrollar un sistema más complejo, se podrían añadir más sensores de temperatura para medir la eficiencia térmica en más de una zona. A parte de medir la eficiencia térmica se podrían añadir distintos tipos de sensores de forma que se hicieran mediciones de eficiencia de otros tipos de dato. Por ejemplo, se podrían añadir sensores de luz para medir los datos obtenidos, dotando a la aplicación de más contenido y haciéndola más atractiva para los consumidores.

Parte 6: Bibliografía y Referencias

6.1 Páginas web

http://www.androidhive.info/2012/05/how-to-connect-android-with-php-mysql/

https://github.com/PhilJay/MPAndroidChart/wiki

http://eclipsesource.com/blogs/2012/10/17/serial-communication-in-java-with-raspberry-pi-and-rxtx/

http://www.weinzierl.de/index.php/en/all-knx/knx-module-en/knx-baos-module-820-en

http://elinux.org/RPi_Serial_Connection

http://www.energylens.com/articles/degree-days

6.2 Referencias

1. http://www.knxcenter.es/node/1
2. https://www.knx.org/es/software/ets/introduccion/index.php?navid=147244147244

6.3 Libros

Para la realización de este trabajo no ha sido necesario el uso de libros. Aun así, se han utilizado los apuntes otorgados de una asignatura cursada en la Universidad Pública de Navarra llamada Desarrollos de Servicios y comunicación en Red de la rama de Telemática del grado en ingeniería en Tecnologías de Telecomunicación, que trata del desarrollo de aplicaciones en Android.

Parte 7: Anexos

7.3 Anexo 1: Parte del protocolo KNX_BAOS module 820 con la información de la generación de las tramas.

The state/length byte is coded as follow:

Bit	Meaning	Value	Description
7	Update flag	0	Value was not updated
		1	Value is updated from bus
6	Read request flag	0	No read request
		1	Read request (GroupValueRead should be sent)
5 - 4	Transmission status	00	Idle/OK
		01	Idle/error
		10	Transmission in progress
		11	Transmission request
3-0	Value length	1-14	Length in bytes of datapoint value

The KNX datapoints with the length less than one byte are coded into the one byte value as follow:

	7	6	5	4	3	2	1	0
1-bit:	0	0	0	0	0	0	0	x
2-bits:	0	0	0	0	0	0	x	x
3-bits:	0	0	0	0	0	x	x	x
4-bits:	0	0	0	0	x	x	x	x
5-bits:	0	0	0	x	x	x	x	x
6-bits:	0	0	x	x	x	x	x	x

2.11. DatapointValue.Ind

This indication is sent asynchronously by the server if the datapoint(s) value is changed and has following format:

Offset	Field	Size	Value	Description
+0	MainService	1	0xF0	Main service code
+1	SubService	1	0xC1	Subservice code
+2	StartDatapoint	1		ID of first datapoint
+3	NumberOfDatapoints	1		Number of datapoints in this indication
+4	First DP ID	1		ID of first datapoint
+5	First DP state/length	1		State/length byte of first datapoint
+6	First DP value	1-14		Value of first datapoint
…	…		…	…
+N-2	Last DP ID	1		ID of last datapoint
+N-1	Last DP state/length	1		State/length byte of last datapoint
+N	Last DP value	1-14		Value of last datapoint

For the coding of the state/length byte see the description of the GetDatapointValue request.

For the coding of the datapoint value see the description of the GetDatapointValue response.

Appendix D. FT1.2 protocol

The FT1.2 transmission protocol is based on the international standard IEC 870-5-1 and IEC 870-5-2 (DIN 19244). As the hardware interface for the transmission is the Universal Asynchronous Receiver Transmitter (UART) used. The frame format for the FT1.2 protocol is fixed to the 8 data bits, 1 stop bit and even parity bit. The default communication speed is 19200 Baud.

D.1. Communication procedure

The typical communication procedure between the host and the *ObjectServer* is shown on figure 7.

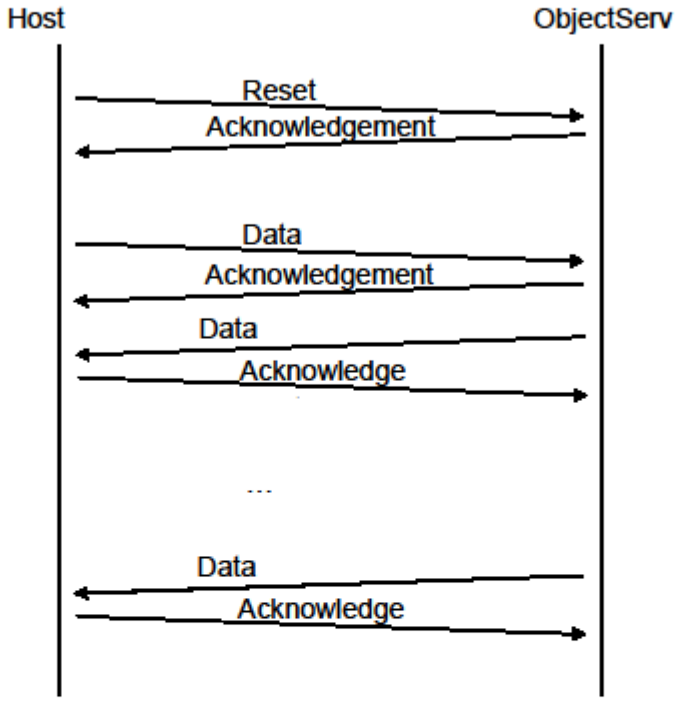


Figure 7. Typical communication procedure

In chapter D.3 is presented an example of the communication between the host and the *ObjectServer*.

D.2. Frame format

Three frame types are defined by the FT1.2 protocol .

The first one is the positiv acknowledgement frame and consists only one byte of the value 0xE5.

The second frame type is 4 bytes length and is used for the reset request and reset indication messages (Fig.8).

Reset.Req:	0x10	0x40	0x40	0x16	Reset.Ind:	0x10	0xC0	0xC0	0x16

Figure 8. Structure of the Reset.Req and Reset.Ind frames

The third frame type is variable length and used for the data messages. The frame structure is presented on figure 9.

0x68	L	L	0x68	CR	data	C	0x16

Figure 9. Structure of the data message

The both fields L contain the length of the data in this frame.

The field CR specifies the control byte of the frame. Its value is 0x73 for all odd frames after reset request sent by the host and 0x53 for the even frames. In the opposite direction (from *ObjectServer* to host) the control byte is 0xF3 for the odd frames and 0xD3 for the even frames.

The field C contains the checksum of the frame and is the arithmetic sum disregarding overflows (modulo 256) over all data and control byte.

D.3. Timings

Parameter	Description	Value
LINE_IDLE_TIMEOUT	Maximal time between two characters in a frame. Line idle detection time (after timeout is expired).	~2 ms
EXCHANGE_TIMEOUT	Maximal time between DATA and ACK frames. Minimal time between two repeated frames.	~30 ms

D.4. Communication example

Host □ *ObjectServer*: Reset Request

{0x10 0x40 0x40 0x16}

ObjectServer □ Client: Acknowledgement

{0xE5}

Host □ *ObjectServer*: GetServerItem.Req (Firmware version)

{0x68 0x05 0x05 0x68 0x73 0xF0 0x01 0x03 0x01 0x68 0x16}

ObjectServer □ Client: Acknowledgement

{0xE5}

ObjectServer □ Client: GetServerItem.Res (Firmare version)

{0x68 0x08 0x08 0x68 0xF3 0xF0 0x81 0x03 0x01 0x03 0x01 0x10 0x7C 0x16}

Host □ *ObjectServer*: Acknowledgement

{0xE5}

Host □ *ObjectServer*: GetServerItem.Req (Serial number)

{0x68 0x05 0x05 0x68 0x53 0xF0 0x01 0x08 0x01 0x4D 0x16}

ObjectServer □ Client: Acknowledgement

{0xE5}

ObjectServer □ Client: GetServerItem.Res (Serial number)

{0x68 0x0D 0x0D 0x68 0xD3 0xF0 0x81 0x08 0x01 0x08 0x06 0x00 0xC5 0x08 0x02 0x00 0x00 0x2A 0x16}

Host □ *ObjectServer*: Acknowledgement

{0xE5}

7.2 Anexo 2: Códigos Java

7.2.1 Clase KNX.java

```
// Relaciona las distintas clases del programa
package KNX;

//clases de la libreria RXTX
        import gnu.io.CommPort;
        import gnu.io.CommPortIdentifier;
        import gnu.io.SerialPort;

        import java.io.FileDescriptor;
        import java.io.IOException;
        import java.io.InputStream;
        import java.io.OutputStream;

        import java.io.DataOutputStream;
        import java.io.InputStreamReader;
        import java.net.HttpURLConnection;
        import java.net.URL;
        import java.net.URLEncoder;
        import java.sql.DriverManager;
        import java.text.DateFormat;
        import java.text.SimpleDateFormat;
        import java.util.Arrays;
        import java.util.Date;
        import java.util.LinkedList;
        import java.util.List;
//clases de la libreria JSON simple 1.1
        import org.json.simple.JSONObject;
        import org.json.simple.JSONValue;



public class KNX
{
    final protected static char[] hexArray =
"0123456789ABCDEF".toCharArray();
    private static Double dato;
    private static int Id;
    private static final String USER_AGENT = "Mozilla/5.0";
    private static final String SERVER_PATH = "http://whattowear.es/";
    public KNX()
    {
        super();
    }

    //COnectar con el puerto serie
    void connect ( String portName ) throws Exception
    {
        CommPortIdentifier portIdentifier =
CommPortIdentifier.getPortIdentifier(portName);
        if ( portIdentifier.isCurrentlyOwned() )
        {
            System.out.println("Error: Port is currently in use");
        }
        else
```

```
        {
            CommPort commPort =
portIdentifier.open(this.getClass().getName(),2000);
            System.out.println(commPort);
            //Si el puerto es puerto serie se introducen los
parametros necesarios para su correcto funcionamiento
            if ( commPort instanceof SerialPort )
            {
                SerialPort serialPort = (SerialPort) commPort;

serialPort.setSerialPortParams(19200,SerialPort.DATABITS_8,SerialPort.
STOPBITS_1,SerialPort.PARITY_EVEN);

                serialPort.disableReceiveTimeout();
                serialPort.enableReceiveThreshold(1);
                //Se recoge los que entre por el puerto serie
                InputStream in = serialPort.getInputStream();
                OutputStream out = serialPort.getOutputStream();
                //Poner a analizar las tramas que se van recibiendo
llamando al metodo que se encarga de ello
                (new Thread(new SerialReader(in,out))).start();

            }
            else
            {
                System.out.println("Error: Only serial ports are
handled by this example.");
            }
        }
    }


    public static class SerialReader implements Runnable
    {
        InputStream in;
        OutputStream out;

        //constructor para recoger las tramas que van llegando
        public SerialReader ( InputStream in, OutputStream out )
        {
            this.in = in;
            this.out = out;
        }

        @Override
        public void run ()
        {
            // Iniciar variable de la clase FrameReader para recoger
los valores analizados de la trama
            FrameReader reader = new FrameReader(in);
            while(true)
            {
                try
                {
                    //Se avanza a la siguiente trama
                    Frame frame = reader.next();
                    this.out.write(0x0E5);
                    System.out.println("\n LA TRAMA LEIDA ES: " +
bytesToHex(frame.trama));
                    System.out.println("\n RECEPCIÓN CORRECTA");
                    //llamamos al metodo que se encarga de enviar los
```

```
datos de interes añadien estos en él
                    sendPost(frame.value,frame.add);


                }
                catch ( IOException e )
                {
                    e.printStackTrace();
                }
            }
        }
    }

    public static void main ( String[] args )
    {
        try
        {
            //Se hace la conexión con el puerto serie de la Raspberry
Pi
            (new KNX()).connect("/dev/ttyAMA0");
        }
        catch ( Exception e )
        {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }

    //Función que se encarga de convertir los bytes que se reciben en
hexadecimal
    public static String bytesToHex(byte[] bytes) {
        char[] hexChars = new char[bytes.length * 2];
        for ( int j = 0; j < bytes.length; j++ ) {
            int v = bytes[j] & 0xFF;
            hexChars[j * 2] = hexArray[v >>> 4];
            hexChars[j * 2 + 1] = hexArray[v & 0x0F];
        }
        return new String(hexChars);
    }

    //Metodo encargado de crear objetos Json con los datos de interes
y hacer las peticiones web
    private static void sendPost(double dato1, int id) {
        // TODO Auto-generated method stub
        dato=dato1;
        Id = id;
        Date date;
        date = new Date();
        //Coger la hora actual y imprimirla por la pantalla en el
formato que queremos
        DateFormat hourFormat = new SimpleDateFormat("HH:mm:ss");
        System.out.println("Hora: "+hourFormat.format(date));
        //Coger la fecha actual y imprimirla por la pantalla en el
formato que queremos
        DateFormat dateFormat = new SimpleDateFormat("yyyy/MM/dd");
        System.out.println("Fecha: "+dateFormat.format(date));
        //Crear un TimeStamp que servira para ordenar las tramas
        DateFormat timestampFormat = new SimpleDateFormat("yyyy/MM/dd
HH:mm:ss");
        System.out.println("Fecha: "+timestampFormat.format(date));
        //Se crea el objeto Json
```

```
        JSONObject jsonObjt= new JSONObject();
        //Se añaden los datos en él
        jsonObjt.put("dato", dato);
        jsonObjt.put("hora", hourFormat.format(date));
        jsonObjt.put("fecha", dateFormat.format(date));
        jsonObjt.put("timestamp", timestampFormat.format(date));
        jsonObjt.put("Id", Id);
        //Imprimir el Json creado por pantalla
        String jsonString = JSONValue.toJSONString(jsonObjt);
        System.out.println("JSON GENERADO:");
        System.out.println(jsonString);
        System.out.println("");

        try {
            //Se genera la URL completa
            String url = SERVER_PATH+"insdatos.php";
            //Se crea un objeto url con la url a la que queremos hacer
la petición
            URL obj = new URL(url);
            //Se crea un objeto de conexión
            HttpURLConnection con = (HttpURLConnection)
obj.openConnection();
            //Se añade la cabecera
            con.setRequestMethod("POST");
            con.setRequestProperty("User-Agent", USER_AGENT);
            con.setRequestProperty("Accept-Language", "en-
US,en;q=0.5");
            //Se crea el parámetro
            String urlParameters = "dato="+jsonString;
            // Hacer la petición POST
            con.setDoOutput(true);
            DataOutputStream wr = new
DataOutputStream(con.getOutputStream());
            wr.writeBytes(urlParameters);
            wr.flush();
            wr.close();
            //Se captura la respuesta del servidor
            int responseCode = con.getResponseCode();
            System.out.println("Post parameters : " + urlParameters);
            System.out.println("Response Code : " + responseCode);

        } catch (Exception e) {
            e.printStackTrace();
        }
    }

}
```

7.2.2 Clase FrameReader.java

```
package KNX;

        import java.io.IOException;
        import java.io.InputStream;
        import java.util.Arrays;



class FrameReader {
    //Variable que almacenara las tramas recibidas
    private final InputStream input;

    //Contructor que recoge las tramas de la clase KNX
    FrameReader(InputStream input) {
        this.input = input;
    }

    Frame next() throws IOException {
        Frame frame = null;
        int b =0;
        byte [] chksum = new byte[20];
        while (frame == null) {

            //Se espera al byte que indica el inicio de la trama
            while(0x068!=input.read()) {
                System.out.println(input.read());
            }
            // Creamos el objeto Frame
            frame = new Frame();
            //Se añade ese byte a la trama del objeto Frame
            frame.trama[0]=(byte)0x68;
            //Se llama al siguiente byte que indica el tamaño de la
trama
            frame.lenght = nextbyte();
            if (frame.lenght!=nextbyte()){
                frame.err=1;
            }
            System.out.println("-----------------------------------
");
            System.out.println("\n Longitud de la trama: " +
frame.lenght);

            frame.trama[1]=(byte)frame.lenght;
            frame.trama[2]=(byte)frame.lenght;
            if (0x068!=nextbyte()){
                frame.err=1;
            }
            frame.trama[3]=(byte)0x68;
            frame.trama[4]=(byte)nextbyte();
            if (0x0F0!=nextbyte()){
                frame.err=1;
            }
            frame.trama[5]=(byte)0xF0;
            if (0x0C1!=nextbyte()){
                frame.err=1;
            }
```

```
            frame.trama[6]=(byte)0xC1;
            //ID del primer data point
            frame.add = nextbyte();
            System.out.println("\n ID: " + frame.add);

            frame.trama[7]=(byte)frame.add;
            //Si el número de data points entregado es mayor de 1
descartamos la trama
            if (0x01!=nextbyte()){
                frame.err=1;
            }
            frame.trama[8]=(byte)0x01;
            //Comprobamos el ID del primer data point entregado
            if (frame.add!=nextbyte()){
                frame.err=1;
            }
            frame.trama[9]=(byte)frame.add;
            //Comprobamos el tamaño de los datos
            b=nextbyte();
            frame.valueSize=(b&0x0F);
            frame.trama[10]=(byte)b;
            System.out.println("\n Tamaño de los datos: " +
frame.valueSize);

            switch (frame.valueSize) {
                //Caso de ser el tamaño de 1 byte
                case 0x01:
                    b=nextbyte();
                    frame.trama[11]=(byte)b;
                    frame.value=b;
                    System.out.println("\n Valor de los datos: " +
frame.value);
                    chksum = Arrays.copyOfRange(frame.trama, 4, 12);
                    b=nextbyte();
                    frame.trama[12]=(byte)b;
                    if((byte)b!=checkSum(chksum)){
                        frame.err=1;
                        System.out.println("\n Checksum calculado es:
"+checkSum(chksum));
                        System.out.println("\n Checksum leido es:
"+b);
                        System.out.println("!!!!!!Error Checksum");
                    }
                    if(nextbyte()!=0x16){
                        frame.err=1;
                        System.out.println("!!!!!!Error Final");
                    }
                    frame.trama[13]=(byte)0x16;
                    break;
                //Caso de ser el tamaño de 2 bytes
                case 0x02:
                    //Se crea una variable que almacene esos 2 bytes
                    byte[] array = new byte[2];
                    b=nextbyte();
                    frame.trama[11]=(byte)b;
                    array[0]=(byte)b;
                    b=nextbyte();
                    frame.trama[12]=(byte)b;
                    array[1]=(byte)b;
                    //Se llama a lafunción que se encarga de convertir
esos bytes en una variable Doble
```

```
                    frame.value=receive2byte(array);
                    System.out.println("\n Valor de los datos: " +
frame.value);
                    chksum = Arrays.copyOfRange(frame.trama, 4, 13);
                    b=nextbyte();
                    frame.trama[13]=(byte)b;
                    if((byte)b!=checkSum(chksum)){
                        System.out.println("\n Checksum calculado es:
"+checkSum(chksum));
                        System.out.println("\n Checksum leido es:
"+b);
                        frame.err=1;
                        System.out.println("!!!!!!Error Checksum");
                    }
                    if(nextbyte()!=0x16){
                        frame.err=1;
                        System.out.println("!!!!!!Error Final");
                    }
                    if(frame.err==1){
                        frame = null;
                    }
                    break;
                //Caso de ser el tamaño de 4 bytes
                case 0x04:
                    //Se crea el Array para almacenar esos 4 bytes
                    byte[] array1 = new byte[4];
                    b=nextbyte();
                    frame.trama[11]=(byte)b;

                    array1[3]=(byte)b;

                    b=nextbyte();
                    array1[2]=(byte)b;
                    frame.trama[12]=(byte)b;

                    b=nextbyte();
                    array1[1]=(byte)b;
                    frame.trama[13]=(byte)b;

                    b=nextbyte();
                    array1[0]=(byte)b;
                    frame.trama[14]=(byte)b;

                    // Convertir esos 4 bytes en un valor Double
                    frame.value += (array1[3] & 0x000000FF) << 24;
                    frame.value += (array1[2] & 0x000000FF) << 16;
                    frame.value += (array1[1] & 0x000000FF) << 8;
                    frame.value += (array1[0] & 0x000000FF);



                    System.out.println("\n Valor de los datos: " +
frame.value);


                    chksum = Arrays.copyOfRange(frame.trama, 4, 15);
                    b=nextbyte();
                    frame.trama[15]=(byte)b;
                    if((byte)b!=checkSum(chksum)){
                        frame.err=1;
                        System.out.println("\n Checksum calculado es:
```

```
"+checkSum(chksum));
                        System.out.println("\n Checksum leido es:
"+b);
                        System.out.println("!!!!!!Error Checksum");
                    }
                    if(nextbyte()!=0x16){
                        frame.err=1;
                        System.out.println("!!!!!!Error Final");
                    }
                    frame.trama[13]=(byte)0x16;
                    break;
                }
            }
            return frame;
        }

        // Función para leer el siguiente byte
        int nextbyte() throws IOException {
            int b = input.read();
            while(b == -1) {
                b = input.read();
            }
            return b;
        }
        //Función para convertir el un array de bytes en un valor double
        public static double receive2byte(byte[] entrada) {

            int exp = ((byte)entrada[0]>>3) & (0x0F);
            int base1 = entrada[0] & (0x07);
            int base = ((int)entrada[1]&0xFF) + (base1 * 256);
            double temp = 0.01 * base * Math.pow(2,exp);

            return temp;
        }
        //Función para añadir el checksum
        public static byte checkSum(byte[] s) {
            byte v = 0;
            for (int z = 0; z < s.length; z++) {
                v = (byte) (v + s[z]);
            }
            return v;
        }


}
```

7.2.3 Clase Frame.java

```
package KNX;
// Clase Frame que sirve para almacenar las variables
class Frame {
    int lenght=0;
    int type=0;
    //Dirección
    int add=0;
    int valueSize=0;
    double value=0;
    int checksum=0;
    byte[] trama = new byte[20];
    int err=0;
}
```

7.3 Anexo 3: Códigos Android

7.3.1 AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.adrian.graficas_datos">
    <uses-permission android:name="android.permission.INTERNET" />
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name"
            android:theme="@style/AppTheme.NoActionBar">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".graficas"
            android:label="@string/graficas"></activity>
        <activity android:name=".calendario"
            android:label="@string/calendario"></activity>
    </application>

</manifest>
```

7.3.2 Build.gradle

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 23
    buildToolsVersion "23.0.3"

    defaultConfig {
        applicationId "com.example.adrian.graficas_datos"
        minSdkVersion 16
        targetSdkVersion 23
        versionCode 1
        versionName "1.0"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-
android.txt'), 'proguard-rules.pro'
        }
    }
}

dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    testCompile 'junit:junit:4.12'
    compile 'com.android.support:appcompat-v7:23.2.1'
    compile 'com.android.support:design:23.2.1'
    compile files('libs/json-simple-1.1.jar')
    compile files('libs/mpandroidchartlibrary-2-1-6.jar')
    compile 'com.loopj.android:android-async-http:1.4.9'

}
```

7.3.3 Layout Main_activity

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true"
    tools:context="com.example.adrian.graficas_datos.MainActivity"
    android:background="#fde3bf">


    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/eficiencia"
        android:layout_centerHorizontal="true"
        android:textSize="50dp"
        android:textAlignment="center"
        android:textColor="#f90606"
        android:textStyle="bold" />

    <ImageView
        android:id="@+id/imbot"
        android:layout_width="150dp"
        android:layout_height="100dp"
        android:src="@drawable/temp"
        android:layout_centerVertical="true"
        android:layout_centerHorizontal="true"
        android:onClick="calendario"/>

</RelativeLayout>
```

7.3.4 Layout calendario

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#fde3bf">

    <TextView
        android:id="@+id/titulo"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/Seleccione_Rango_de_Fechas"
        android:textAlignment="center"
        android:textSize="40dp"
        android:textStyle="bold"
        android:textColor="#f90606" />
<TextView
    android:layout_marginTop="40dp"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/Desde"
    android:textStyle="bold"
    android:textSize="30dp"
    android:textColor="#f90606"
    android:id="@+id/textView" />
    <TextView
        android:id="@+id/fechaInicio"
        android:layout_width="fill_parent"
        android:layout_height="20dp"
        android:background="#f9f9f9"
        android:textAlignment="center"/>
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/Seleccionar"
        android:id="@+id/buttonIn"
        android:textAlignment="center"
        android:onClick="dialog1"/>

    <TextView
        android:layout_marginTop="40dp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/Hasta"
        android:textStyle="bold"
        android:textSize="30dp"
        android:textColor="#f90606"
       />
    <TextView
        android:id="@+id/fechaFinal"
        android:layout_width="fill_parent"
        android:layout_height="20dp"
        android:background="#f9f9f9"
        android:textAlignment="center"
        />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
            android:text="@string/Seleccionar"
            android:id="@+id/buttonFi"
            android:textAlignment="center"
            android:onClick="dialog2"/>
        <RelativeLayout
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="50dp">
            <Button
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_centerHorizontal="true"
                android:layout_centerVertical="true"
                android:text="@string/graficar"
                android:onClick="graficar"/>
        </RelativeLayout>

</LinearLayout>
```

7.3.5 Layout gráficas

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent">

    <com.github.mikephil.charting.charts.LineChart
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:id="@+id/temperaturas">

    </com.github.mikephil.charting.charts.LineChart>
    <com.github.mikephil.charting.charts.BarChart
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:id="@+id/consumo">

    </com.github.mikephil.charting.charts.BarChart>

</LinearLayout>
```

7.3.6 Clase MainActivity

```
package com.example.adrian.graficas_datos;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.Menu;
import android.view.MenuItem;

public class MainActivity extends Activity {

    //Método al que se llama al acceder el activity
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it
is present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }
    //Método al que se le llama al pulsar el botón de este layout
    public void calendario(View v){
        Intent intent=new Intent(this,calendario.class);
        startActivity(intent);
    }
}
```

7.3.7 Clase calendario

```
package com.example.adrian.graficas_datos;

import android.app.Activity;
import android.app.DatePickerDialog;
import android.app.Dialog;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.CalendarView;
import android.widget.DatePicker;
import android.widget.TextView;
import android.widget.Toast;

import java.util.Calendar;
import java.util.Date;

/**
 * Created by Adrian on 14/06/2016.
 */
public class calendario extends Activity {

    String fechaInicio,fechaFinal,fiEsp,ffEsp;
    int year,month,day;
    public static int DIALOG1_ID=1,DIALOG2_ID=2;
    int cont=1;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.calendario);
        //Se crea el objeto que almacena la fecha
        final Calendar cal= Calendar.getInstance();
        year=cal.get(Calendar.YEAR);
        month=cal.get(Calendar.MONTH);
        day=cal.get(Calendar.DAY_OF_MONTH);
    }
  //Métodos llamados al pulsar los botones de escoger fecha
  public void dialog1(View v){
      showDialog(DIALOG1_ID);
  }
    public void dialog2(View v){
        showDialog(DIALOG2_ID);
    }

    //Abre el dialogo con los calendarios
    @Override
    protected Dialog onCreateDialog(int id){
        if(id==DIALOG1_ID)
            return new
DatePickerDialog(this,dpickerListener,year,month,day);
        if(id==DIALOG2_ID)
            return new
DatePickerDialog(this,dpickerListener1,year,month,day);
        return null;
    }

    // Guarda la fecha inicial escogida y lo hace visible en el
textview
```

```
    private DatePickerDialog.OnDateSetListener dpickerListener
            =new DatePickerDialog.OnDateSetListener(){
        @Override
        public void onDateSet(DatePicker view,int year,int
monthofyear,int dayofmonth){
            TextView textView= (TextView)
findViewById(R.id.fechaInicio);
            year=year;
            month=monthofyear+1;
            day=dayofmonth;
            if (day<10 && month<10) {
                fechaInicio = year + "-" + "0" + month + "-" + "0" +
day;
                fiEsp="0"+day+"/"+"0"+month+"/"+year;
            }else if(day<10 && month>=10){
                fechaInicio = year + "-" + month + "-" + "0" + day;
                fiEsp="0"+day+"/"+month+"/"+year;
            }else if(day>=10 && month<10){
                fechaInicio = year + "-" + "0"+month + "-" + day;
                fiEsp=day+"/"+"0"+month+"/"+year;
            }else {
                fechaInicio = year + "-" +month + "-" + day;
                fiEsp=day+"/"+month+"/"+year;
            }
            textView.setText(fiEsp);
            month=monthofyear;
        }


    };

    // Guarda la fecha final escogida y lo hace visible en el textview
    private DatePickerDialog.OnDateSetListener dpickerListener1
            =new DatePickerDialog.OnDateSetListener(){
        @Override
        public void onDateSet(DatePicker view,int year,int
monthofyear,int dayofmonth){
            TextView textView= (TextView)
findViewById(R.id.fechaFinal);
            year=year;
            month=monthofyear+1;
            day=dayofmonth;
            if (day<10 && month<10) {
                ffEsp="0"+day+"/"+"0"+month+"/"+year;
            }else if(day<10 && month>=10){
                ffEsp="0"+day+"/"+month+"/"+year;
            }else if(day>=10 && month<10){
                ffEsp=day+"/"+"0"+month+"/"+year;
            }else {
                ffEsp=day+"/"+month+"/"+year;
            }
            //Se avanza el día para realizar el filtrado
posteriormente
            day=day+1;
            if (day<10 && month<10) {
                fechaFinal = year + "-" + "0" + month + "-" + "0" +
day;
            }else if(day<10 && month>=10){
                fechaFinal = year + "-" + month + "-" + "0" + day;
            }else if(day>=10 && month<10){
                fechaFinal = year + "-" + "0"+month + "-" + day;
```

```
            }else {
                fechaFinal = year + "-" +month + "-" + day;
            }
            textView.setText(ffEsp);
            month=monthofyear;

        }


    };

    //Método al que se le llama al pusar el boton ver gráficas
    public void graficar(View v) {
        TextView t1= (TextView) findViewById(R.id.fechaInicio);
        TextView t2= (TextView) findViewById(R.id.fechaFinal);
        //Se analiza si estan bien escogidas las fechas
        if(t1.getText()!="" && t2.getText()!="") {
            String fec1= (String) t1.getText();
            String fec2= (String) t2.getText();
            String mes1= fec1.substring(3,5);
            String mes2= fec2.substring(3,5);
            if (mes1.equals(mes2)) {
                String dia1=fec1.substring(0,2);
                String dia2=fec2.substring(0,2);
                int diaIn=Integer.parseInt(dia1);
                int diaFi=Integer.parseInt(dia2);
                if (diaIn<=diaFi) {
                    //Se llama a la siguiente clase y se añaden los
valores que se necesitan en ella
                    Intent intent = new Intent(this, graficas.class);
                    intent.putExtra("fechaInicio", fechaInicio);
                    intent.putExtra("fechaFinal", fechaFinal);
                    startActivity(intent);
                }else {
                    Toast.makeText(getApplicationContext(),"Periodo de
tiempos mal seleccionado, cambie el orden de las fechas",
Toast.LENGTH_SHORT).show();
                }
            }else {
                Toast.makeText(getApplicationContext(),"Las fechas
deben de ser del mismo mes", Toast.LENGTH_SHORT).show();
            }
        }else {
            Toast.makeText(getApplicationContext(),"Seleccione las
fechas", Toast.LENGTH_SHORT).show();
        }
    }

}
```

7.3.8 Clase Gráficas

```
package com.example.adrian.graficas_datos;

import android.app.Activity;
import android.content.Intent;
import android.graphics.Color;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.Toast;

import com.github.mikephil.charting.charts.BarChart;
import com.github.mikephil.charting.charts.LineChart;
import com.github.mikephil.charting.data.BarData;
import com.github.mikephil.charting.data.BarDataSet;
import com.github.mikephil.charting.data.BarEntry;
import com.github.mikephil.charting.data.Entry;
import com.github.mikephil.charting.data.LineData;
import com.github.mikephil.charting.data.LineDataSet;
import com.loopj.android.http.AsyncHttpClient;
import com.loopj.android.http.AsyncHttpResponseHandler;
import com.loopj.android.http.RequestParams;

import org.json.JSONArray;

import java.util.ArrayList;
import java.util.Date;

import cz.msebera.android.httpclient.Header;

/**
 * Created by Adrian on 05/06/2016.
 */
public class graficas extends Activity {
    private ArrayList<String> datostemp;
    private ArrayList<String> datoscons;
    private ArrayList<String> fechatemp;
    private ArrayList<String> fechatemp1;
    private ArrayList<String> fechacons;
    private ArrayList<String> idtemp;
    private ArrayList<String> datostempint;
    private ArrayList<String> datostempext;
    private ArrayList<String> fechatempint;
    private ArrayList<String> fechatempext;

    private String fechaInicio,fechFinal,tot;
    private int rango,cont=0;
    private Boolean existeIn=true,existeFi=true;
    private Date dateIn,dateFi;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.graficas);
        //Coger los valores de la clase anterior
        fechaInicio= getIntent().getStringExtra("fechaInicio");
        fechFinal= getIntent().getStringExtra("fechaFinal");
        //Llamar al metodo que hace la petición web
```

```
        conseguirdatos();
    }

    //Método que se encarga de hacer la petición web
    private void conseguirdatos() {
        //Crear el objeto cliente
        AsyncHttpClient client= new AsyncHttpClient();
        String url = "http://whattowear.es/recogdatos.php";
        RequestParams parametros= new RequestParams();
        // Hacer el POST
        client.post(url, parametros, new AsyncHttpResponseHandler() {
            @Override
            public void onSuccess(int statusCode, Header[] headers,
byte[] responseBody) {
                if (statusCode == 200) {
                    //Se llama al método encargado de organizar las
respuestas
                    organizarRespuestas(obtDatosJSONdat(new
String(responseBody)));
                }
            }

            @Override
            public void onFailure(int statusCode, Header[] headers,
byte[] responseBody, Throwable error) {

            }
        });
    }

    //Método que se encarga de organizar las respuestas
    //y almacenarlas en sus correspondientes listas
    private void organizarRespuestas(ArrayList<String> respuestas) {
        datostemp=new ArrayList<String>();
        datoscons=new ArrayList<String>();
        fechatemp=new ArrayList<String>();
        fechatemp1=new ArrayList<String>();
        fechacons=new ArrayList<String>();
        idtemp=new ArrayList<String>();

        //Hacer el ajuste de fechas por si no hay datos en las fechas
seleccionadas
        while (cont<=30) {
            for (int j = 3; j < respuestas.size(); j = j + 4) {
                String comprobarfecha = respuestas.get(j);
                if (comprobarfecha.equals(fechaInicio)) {
                    existeIn = false;
                } else if (comprobarfecha.equals(fechFinal)) {
                    existeFi = false;
                }
            }
            if (existeIn) {
                String dia = fechaInicio.substring(8, 10);
                int dias = Integer.parseInt(dia);
                dias++;
                String mediafecha = fechaInicio.substring(0, 8);
                if (dias < 10) {
                    fechaInicio = mediafecha + "0" + dias;
                } else {
                    fechaInicio = mediafecha + dias;
                }
```

```
            }
            if (existeFi) {
                String dia = fechFinal.substring(8, 10);
                int dias = Integer.parseInt(dia);
                dias++;
                String mediafecha = fechFinal.substring(0, 8);
                if (dias < 10) {
                    fechFinal = mediafecha +"0"+ dias;
                } else {
                    fechFinal = mediafecha + dias;
                }
            }
            if(!existeFi&&!existeIn){
                break;
            }
            cont++;
        }
        //Ver el id de los datos y almacenar en las listas
        for(int i=2;i<respuestas.size();i=i+4){
            String Id = respuestas.get(i);
            String fecha = respuestas.get(i+1);
            if (Id.equals("2") || Id.equals("3")){
                if (fecha.equals(fechFinal)) {rango=0;}
                if (fecha.equals(fechaInicio)) {rango=1;}
                if (rango==1) {
                    datostemp.add(respuestas.get(i - 2));
                    fechatemp.add(respuestas.get(i - 1));
                    fechatemp1.add(respuestas.get(i+1));
                    idtemp.add(Id);

                }
            }
            if (Id.equals("4")){
                if (fecha.equals(fechFinal)) {rango=0;}
                if (fecha.equals(fechaInicio)) {rango=1;}
                if (rango==1) {
                    datoscons.add(respuestas.get(i - 2));
                    fechacons.add(respuestas.get(i + 1));

                }
            }
        }
        //Llamar a los métodos que hacen las gráficas
        tempgraf(datostemp, fechatemp, idtemp);
        consgraf(datoscons, fechacons,datostemp,fechatemp1,idtemp);
    }

    //Recoge los objetos Json y almacena estos en una lista de Strings
ordenada
    public ArrayList<String> obtDatosJSONdat(String response) {

        ArrayList<String> respuestaList= new ArrayList<String>();

        try {
            JSONArray json= new JSONArray(response);
            String respuesta;

            for (int i=0; i<json.length();i++){
                respuesta = json.getJSONObject(i).getString("datos");
                respuestaList.add(respuesta);
                respuesta = json.getJSONObject(i).getString("hora");
```

```
                respuestaList.add(respuesta);
                respuesta = json.getJSONObject(i).getString("id");
                respuestaList.add(respuesta);
                respuesta = json.getJSONObject(i).getString("fecha");
                respuestaList.add(respuesta);
            }

        } catch (Exception e) {
            // TODO: handle exception
            e.printStackTrace();
        }
        return respuestaList;
    }
    //Metodo encargado de hacer la gráfica del consumo
    private void consgraf(ArrayList<String> datoscons,
ArrayList<String> fechacons, ArrayList<String> datostemp,
ArrayList<String> fechatemp, ArrayList<String> idtemp) {
        BarChart barchart = (BarChart) findViewById(R.id.consumo);
        ArrayList<String> datint=new ArrayList<String>();
        ArrayList<String> fechaint=new ArrayList<String>();
        ArrayList<String> datext=new ArrayList<String>();
        ArrayList<String> fechaext=new ArrayList<String>();
        ArrayList<Double> valint=new ArrayList<Double>();
        ArrayList<Double> valext=new ArrayList<Double>();
        ArrayList<Double> valcons=new ArrayList<Double>();
        ArrayList<BarEntry> entries = new ArrayList<>();
        ArrayList<BarEntry> gdentries = new ArrayList<>();

        //Distinguir los datos de la temperatura
        for(int i=0;i<datostemp.size();i++){
            if (this.idtemp.get(i).equals("2")) {
                datint.add(datostemp.get(i));
                fechaint.add(fechatemp.get(i));
            }else if (this.idtemp.get(i).equals("3")){
                datext.add(datostemp.get(i));
                fechaext.add(fechatemp.get(i));
            }
        }

        //Promedio diario de los datos de temperatura interior
        String fianterior= fechaint.get(0);
        double totalint=0;
        int cant=0;
        Double valorint;
        for (int j=0;j<fechaint.size();j++){
            if(fechaint.get(j).equals(fianterior)){
                double datoint=Double.parseDouble(datint.get(j));
                totalint=totalint+datoint;
                cant++;
            }else{
                valorint=totalint/cant;
                valint.add(valorint);
                fianterior=fechaint.get(j);
                double datoint=Double.parseDouble(datint.get(j));
                totalint=datoint;
                cant=1;
            }
        }
        valorint=totalint/cant;
        valint.add(valorint);
```

```
        //Promedio diario de los datos de temperatura exterior
        String feanterior= fechaext.get(0);
        double totalext=0;
        int cantext=0;
        Double valorext;
        for (int j=0;j<fechaext.size();j++){
            if(fechaext.get(j).equals(feanterior)){
                double datoext=Double.parseDouble(datext.get(j));
                totalext=totalext+datoext;
                cantext++;
            }else{
                valorext=totalext/cantext;
                valext.add(valorext);
                feanterior=fechaext.get(j);
                double datoext=Double.parseDouble(datext.get(j));
                totalext=datoext;
                cantext=1;
            }
        }
        valorext=totalext/cantext;
        valext.add(valorext);

        //Cálculo del consumo diario
        String anterior = fechacons.get(0);
        double total=0;
        double totant=0;
        int cont=0;
        double valor=0;
        for(int i=0;i<datoscons.size();i++){
            if (fechacons.get(i).equals(anterior)) {
                double dato=Double.parseDouble(datoscons.get(i));
                total=dato;
            }else {
                valor=total-totant;
                valcons.add(valor);
                tot= String.valueOf(valor);
                entries.add(new BarEntry(Float.valueOf(tot), cont));
                cont++;
                totant=total;
                total=Double.parseDouble(datoscons.get(i));
            }
        }
        valor=total-totant;
        valcons.add(valor);
        tot= String.valueOf(valor);
        entries.add(new BarEntry(Float.valueOf(tot), cont));

        //Hacer la normalización gradosdía
        for (int i=0;i<valint.size();i++){
            Double graddia=(valint.get(i)-4)-valext.get(i);
            if (graddia>0){
                double kWhgd= valcons.get(i)/graddia;
                String valkWhgd= String.valueOf(kWhgd);
                gdentries.add(new
BarEntry(Float.valueOf(valkWhgd),i));
            }else {
                double kWhgd=valcons.get(i);
                String valkWhgd= String.valueOf(kWhgd);
                gdentries.add(new
BarEntry(Float.valueOf(valkWhgd),i));
            }
```

```
        }

        //Añadir los valores resultantes a las gráficas
        BarDataSet cons = new BarDataSet(entries, "kWh");
        cons.setColor(Color.BLUE);
        BarDataSet consgd = new BarDataSet(gdentries, "kWh/gd");
        consgd.setColor(Color.GREEN);
        ArrayList<BarDataSet> setcons=new ArrayList<>();
        setcons.add(cons);
        setcons.add(consgd);
        ArrayList<String> Xaxis = new ArrayList<String>();
        Xaxis.add(fechacons.get(0));
        for (int j=1;j<fechacons.size();j++) {
            if (!fechacons.get(j).equals(fechacons.get(j-1))) {
                Xaxis.add(fechacons.get(j));
            }
        }
        //Hacer la gráfica
        BarData data = new BarData(Xaxis, setcons);
        barchart.setGridBackgroundColor(Color.BLACK);
        barchart.setData(data);
        barchart.setDescription("Consumo");
        barchart.setDescriptionColor(Color.WHITE);
        barchart.moveViewToX(0);
    }

    private void tempgraf(ArrayList<String> datostemp,
ArrayList<String> fechatemp, ArrayList<String> idtemp) {


        LineChart lineChart = (LineChart)
findViewById(R.id.temperaturas);
        ArrayList<Entry> entriesint = new ArrayList<>();
        ArrayList<Entry> entriesext = new ArrayList<>();

        //Distinguir los datos de la temperatura
        for(int i=0;i<datostemp.size();i++){
            if (this.idtemp.get(i).equals("2")) {
                entriesint.add(new
Entry(Float.valueOf(datostemp.get(i)), i));
            }else if (this.idtemp.get(i).equals("3")){
                entriesext.add(new
Entry(Float.valueOf(datostemp.get(i)), i));
            }
        }


        //Meter los datos en las gráficas y modificarlas
        LineDataSet tempint = new LineDataSet(entriesint, "Temp.
Interrior");
        tempint.setDrawCubic(true);
        tempint.setColor(Color.RED);
        tempint.setCircleColor(Color.RED);
        tempint.setValueTextColor(Color.RED);
        tempint.setFillColor(Color.RED);
        tempint.setDrawFilled(false);
        LineDataSet tempext = new LineDataSet(entriesext, "Temp.
Exterior");
        tempext.setDrawCubic(true);
        tempext.setColor(Color.YELLOW);
        tempext.setValueTextColor(Color.YELLOW);
```

```
        tempext.setCircleColor(Color.YELLOW);
        tempext.setFillColor(Color.YELLOW);
        tempext.setDrawFilled(false);
        ArrayList<String> Xaxis = new ArrayList<String>();
        for (int j=0;j<fechatemp.size();j++) {
                Xaxis.add(fechatemp.get(j));
        }

        //Hacer la gráfica
        ArrayList<LineDataSet> lines= new ArrayList<LineDataSet>();
        lines.add(tempint);
        lines.add(tempext);
        LineData data= new LineData(Xaxis,lines);
        lineChart.setGridBackgroundColor(Color.BLACK);
        lineChart.setData(data);
        lineChart.setDescription("°C");
        lineChart.setDescriptionColor(Color.WHITE);
        lineChart.moveViewToX(0);

    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it
is present.
        getMenuInflater().inflate(R.menu.graficas, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }
}
```