Trabajo de Fin de Grado

Controlador central para un sistema domótico utilizando el protocolo inalámbrico ZigBee

Central controller for a home automation system using the Zigbee wirless protocol

Enrique Hernández Bello

La Laguna, 6 de septiembre de 2016

D. **Alberto Francisco Hamilton Castro**, con N.I.F. 43.773.884-X profesor Titular de Universidad adscrito al Departamento de Ingeniería Informática y Sistemas de la Universidad de La Laguna, como tutor

C E R T I F I C A

Que la presente memoria titulada:

"Controlador central para un sistema domótico utilizando el protocolo inalámbrico ZigBee"

ha sido realizada bajo su dirección por D. **Enrique Hernández Bello**, con N.I.F. 78.616.477-X.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 6 de septiembre de 2016

Agradecimientos

Al profesor Alberto Hamilton, tutor de este proyecto, por su condescendencia conmigo durante su realización.

A «Albertito» y a «Migue», quienes me acompañaron en la recta final.

A mi familia, amigos, compañeros y profesores.

Y en especial, a MIS PADRES. Pues sin su apoyo, no hubiera mantenido mi decisión de llegar hasta aquí.

Licencia

Resumen

El Internet de las Cosas (IoT) es un concepto cada vez más familiar en nuestros hogares, fruto de conectar cada día, más y más dispositivos y electrodomésticos a la red de redes.

Con el auge cada vez más alto de dispositivos a interconectar, se hace necesario un sistema central que permita gestionarlos de forma flexible y eficiente.

Este proyecto lleva a cabo la confección de un sistema domótico de control remoto basado en tecnologías específicas, entre ellas, el protocolo de comunicación ZigBee.

Para su implementación, hace uso de módulos XBee integrando, además, otros diferentes componentes que aportan cada funcionalidad marcada como requisito.

Entre las muchas características de este proyecto, podemos destacar las siguientes:

- *Es de filosofía libre y ha servido para contribuir a mejorar proyectos libres ya existentes.*
- *A destacar su diseño modular, que permite obtener las mejores características de cada componente y facilita la evolución del sistema intercambiándolos por otros equivalentes cuando sea necesario.*
- *Está pensado para ser distribuido en un Sistema Embebido, como es*

la PandaBoard.

- *Ha sido diseñado para maximizar la flexibilidad en su configuración para adaptarse de la mejor forma a las necesidades del usuario.*

Palabras clave: Domótica, IoT, XBee, ZigBee, PandaBoard, Ubuntu Core, Domoticz, Python, MQTT, Node-RED, Sensores, Actuadores.

Abstract

The Internet of Things (IoT) is an increasingly familiar concept in our homes, the result of connecting each day, more and more devices and appliances to the network of networks.

With the increasingly high interconnect devices, we need a central system to manage in a flexibly and efficiently way.

This project carries out the making of a home automation remote control system based on specific technologies, including ZigBee communication protocol.

For its implementation, uses XBee modules and also integrate other components that provide different functionality marked as requirement.

Among the many features of this project, we highlight the following:

- *Promotes the «libre» philosophy and it has helped to improve another existing open source projects.*
- *To highlight its modular design. Thanks to him, you can get the best features of each component and eases the evolution of the system, allowing the exchange them for other equivalent when necessary.*
- *It is intended to be distributed as an embedded system, such as the PandaBoard.*
- *It has been designed to maximize the flexibility to fit in the best way to user needs.*

Keywords: Home Automation, IoT, XBee, ZigBee, PandaBoard, Ubuntu Core, Domoticz, Python, MQTT, Node-RED, Sensors, Actuators.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1
Introducción

1.1 Antecedentes

La evolución de la tecnología ha llegado a un punto en el que cualquier objeto puede tener una interfaz electrónica que lo controle. El siguiente paso lógico a conseguir, es que esos objetos cotidianos, convertidos en dispositivos electrónicos, sean capaces de comunicarse para que trabajen de forma colaborativa entre sí.

Dado el caso, para conseguir que un gran número de dispositivos funcione de manera coherente, es necesario un sistema de control que gestione sus modos de funcionamiento, su relación con los eventos diarios y además monitorice cuál es su estado actual.

1.2 Objetivos del proyecto

En términos generales, se propone configurar un miniordenador host de tipo PandaBoard, para que sea capaz de interconectar todos los dispositivos inalámbricos huéspedes disponibles basados en módulos XBee.

A continuación, se listan otros objetivos que debe cumplir el proyecto:

- El sistema debe quedar sobre alguna distribución GNU/Linux

instalada y configurada en la placa de desarrollo PandaBoard.

- Para emitir y recibir mensajes sobre el protocolo ZigBee, un módulo XBee en modo coordinador, debe quedar conectado electrónicamente y de manera fija a la placa de desarrollo PandaBoard.
- El sistema debe poderse comunicar y configurar otros dispositivos XBee sensores/actuadores de funcionalidad genérica.
- Debe disponer de una interfaz de usuario, que permita su puesta en funcionamiento y se pueda presentar como una solución lo más completa y usable posible.
- Además, se consideraría completar su funcionalidad añadiendo un control remoto y haciendo uso de una pantalla táctil o salida HDMI para mostrar información sobre el estado del sistema.

Capítulo 2
Conocimientos previos

Después de haber introducido las bases del proyecto en el capítulo anterior, ahora es el turno de describir en mayor o menor medida, las tecnologías relacionadas con el mismo.

2.1 PandaBoard

La **PandaBoard** es un ordenador de placa reducida de bajo coste y bajo consumo que, como plataforma de desarrollo, está basada en el «System on a Chip (SoC)» Texas Instruments OMAP4430 o superior.

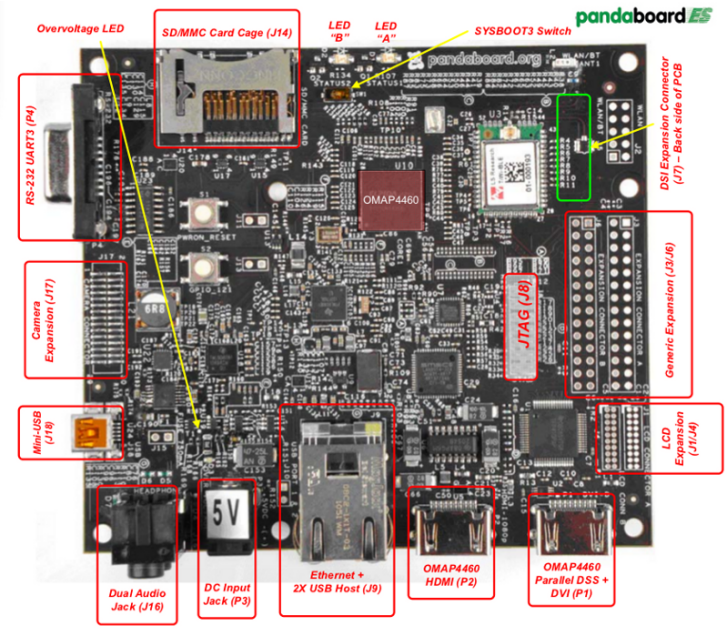


Figura 2.1: Vista descriptiva de la PandaBoard ES

Esta placa está disponible en el mercado desde octubre del año 2010 y, a pesar que en su momento fue una placa de desarrollo soportada por la comunidad, desde el año 2012, ya no lo está, quedando como obsoleta frente a otras placas de desarrollo más novedosas como Raspberry-PI.

La versión más reciente de PandaBoard fue la PandaBoard ES, que contaba con el SoC OMAP4460, cuya principal mejora, eran el incremento de velocidad en CPU y GPU.

Concretamente, esta última versión ha sido con la que se ha contado para este proyecto. Algunas de las características más destacadas que describen esta placa de desarrollo, son:

- Arquitectura ARMv7-A multicore de 32 bits
- CPU ARM Cortex-A9 Dual-Core a 1,2Ghz
- GPU PowerVR SGX540 que provee una salida HDMI de hasta 1080p
- 1GB de memoria RAM DDR2
- Soporte para tarjetas SDHC y puertos USB
- Conectividad LAN, Bluetooth y WLAN 802.11bgn.

2.2 Protocolo Zigbee

ZigBee es un protocolo de comunicaciones inalámbrico creado a principios de siglo por la empresa ZigBee Alliance, cuyo objetivo era ampliar el estándar IEEE 802.15.4 para cumplir con las siguientes características:

- Bajo coste en cuanto al despliegue y mantenimiento de la red.
- Velocidad de transmisión reducida.

- Bajo consumo de potencia por parte de los dispositivos, lo que permite prolongar el uso de las baterías.
- Instalación fácil y sencilla.
- La creación de redes ampliables y flexibles.

2.2.1 Tipos de dispositivos

Las redes 802.15.4 y en concreto ZigBee, definen tres tipos de elementos según la función desempeñada en la red:

- **Coordinador:** Debe haber un único elemento de este tipo en la red. Es el encargado de crear, inicializar y controlar la red, estableciendo aspectos como el canal de comunicaciones y asignando parámetros como las direcciones de red. Una vez concluidas estas funciones, su comportamiento pasa a ser el mismo que la de un Router.
- **Router:** Son nodos de funcionalidad completa que se encargan de interconectar dispositivos encaminando los mensajes hacia el nodo destino. Se emplean para extender la cobertura de red y crear nuevos caminos que minimicen la congestión o sobrevivan a la caída de algún nodo.
- **Nodo final**: Su funcionalidad se reduce a la necesaria para comunicarse con el nodo padre (coordinador o router) pasando a estar dormido la mayor parte del tiempo y, por lo tanto, reduciendo el consumo de energía.

2.3 XBee

XBee es la familia de módulos de radio pertenecientes a la compañía Digi International que, haciendo uso del protocolo ZigBee brindan un medio inalámbrico para la interconexión y comunicación entre dispositivos.

Figura 2.2: XBee Series2 U.FL

Hay una gran variedad de módulos XBee, aunque básicamente se pueden agrupar en dos tipos, los de la Serie 1 y los de la Serie 2. Ambos son muy similares e incluso tienen las mismas dimensiones, sin embargo, no son compatibles entre sí.

	XBee Serie1	**XBee Serie2**
Alcance interior	Hasta 30m	Hasta 40m
Potencia sin obstáculos	Hasta 100m	Hasta 120m
Potencia de transmisión	1mW (0dbm)	2mW (+3dbm)
Tensión de alimentación	2.8 – 3.4V	2.8 – 3.6V
Sensibilidad del receptor	-92dbm (1% PER)	-98dbm (1% PER)

Corriente de apagado	10 uA	1 uA
Topologías de red	Punto a punto y estrella	Punto a punto, estrella y malla
Rango de temperatura	-40 a 85C	-40 a 85C
Nº de Entradas/Salidas digitales	8	10
Nº de entradas analógicas	7	4
Nº de salidas analógicas	2	Ninguna

Tabla 2.1: Resumen de tipos de XBee

En este proyecto se cuenta con módulos XBee de la Serie2 cuyo identificador es el: XB24-BUIT-004.

Uno de ellos, cuya función estará definida por el *modo coordinador*, irá directamente conectado a la placa de desarrollo PandaBoard.

El otro, cuya función estará definida por el *modo de nodo final*, es el que realizará las funciones de sensor y/o actuador, dependiendo de lo que tenga conectado a cada uno de sus pines.

2.3.1 Pines del módulo

Según lo descrito en la figura 2.3, cada módulo dispone de 20 terminales para diferentes propósitos. La función de los pines es configurable. Dependiendo de dicha configuración, el pin puede realizar, entre otras, tareas de entrada y salida, digitales o analógicas.

Pin #	Name	Direction	Default State	Description
1	VCC	-	-	Power Supply
2	DIO13/DOUT	Both	Output	UART Data out
3	DIO14/DIN/nCONFIG	Both	Input	UART Data In
4	DIO12/SPI_MISO	Both	Disabled	GPIO/ SPI slave out
5	nRESET	Input	Input	Module Reset
6	DIO10/RSSI PWM/PWM0	Both	Output	RX signal strength indicator/GPIO
7	DIO11/PWM1	Both	Disabled	GPIO
8	reserved	-	-	Do Not Connect
9	DIO8/nDTR/SLEEP_RQ	Both	Input	Pin Sleep Control line /GPIO
10	GND	-	-	Ground
11	DIO4/SPI_MOSI	Both	Disabled	GPIO/SPI slave In
12	DIO7/nCTS	Both	Output	Clear-to-Send Flow Control/GPIO
13	DIO9/ON_nSLEEP	Both	Output	Module Status Indicator/GPIO
14	VREF	-	-	Not connected
15	DIO5/ASSOCIATE	Both	Output	Associate Indicator/GPIO
16	DIO6/nRTS	Both	Input	Request-to-Send Flow Control/GPIO
17	DIO3/AD3 /SPI_nSSEL	Both	Disabled	Analog Input/GPIO/SPI Slave Select
18	DIO2/AD2 /SPI_CLK	Both	Disabled	Analog Input/GPIO/SPI Clock
19	DIO1/AD1 /SPI_nATTN	Both	Disabled	Analog Input/GPIO/SPI Attention
20	DIO0/AD0/CB	Both	Disabled	Analog Input/Commissioning Button/GPIO

Figura 2.3: Asignación de pines en XBee

Como se puede ver en la figura 2.4, para la realización de las pruebas de conexión necesarias, se ha trabajado con el módulo XBee del nodo final, conectado a una placa protoboard para disponer de las siguientes funcionalidades:

- Un botón pulsador conectado a un pin digital de entrada.

- Un potenciómetro conectado a un pin analógico de entrada.
- Y un LED conectado a un pin digital de salida.

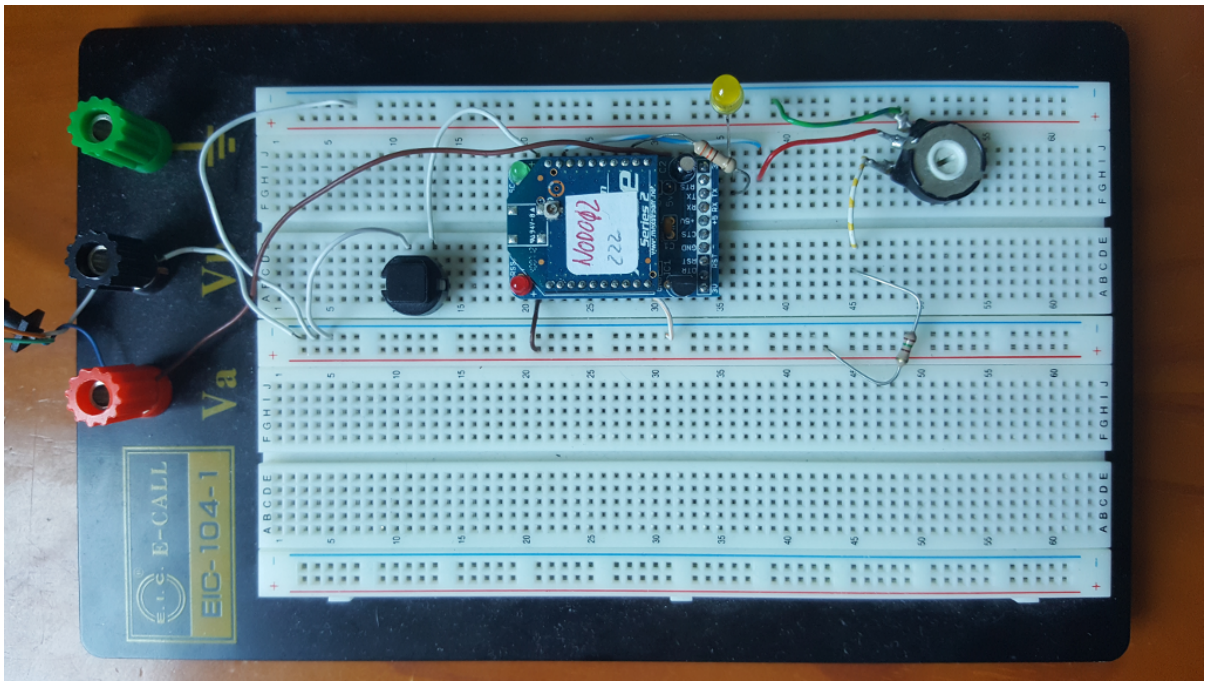


Figura 2.4: Tabla de asignación de pines XBee Serie2

La conexión de los XBee en los nodos finales es directa y sin ningún tipo de microcontrolador como Arduino de por medio, lo que hubiera facilitado la gestión de los mismos usando, por ejemplo, el feedback de la comunidad MySensors.

La lógica de la gestión de los nodos finales recae, entonces, únicamente en el controlador.

2.3.2 Modo de comunicación

Los módulos XBee disponen de dos modos de trabajo -seleccionables a través de parámetros de configuración- para realizar las comunicaciones:

- **Modo transparente:** Es la manera más sencilla de comunicar módulos XBee, pues se configura una conexión de tipo serial en la que

todo carácter recibido por el pin Rx (DIN) es enviada al pin Tx (DOUT) del destino deseado. En este modo es posible realizar tanto comunicaciones *punto a punto* como *punto a multipunto*.

- **Modo API (Application Programming Interface):** Este modo, que es el que nos interesa, tiene como objetivo transmitir los datos de una manera segura y estructurada, empaquetada en tramas (frames) que definen operaciones y eventos dentro del módulo.

2.3.3 Comandos básicos del Modo API

Para la comunicación con los módulos, se hace uso de instrucciones AT asociadas a un frame específico. En el documento de referencia del módulo se puede encontrar información detallada de estos comandos, no obstante, a continuación se recopila un resumen de los más relevantes.

- **DX**: Lectura/escritura de la configuración de un PIN de entrada/salida definidas como DIOx. De forma opcional, puede especificarse alguno de los siguientes parámetros de configuración:

Pin Command Parameter	Description
0	Unmonitored digital input
1	Reserved for pin-specific alternate functionalities
2	Analog input, single ended (A/D pins only)
3	Digital input, monitored
4	Digital output, default low
5	Digital output, default high
6-9	Alternate functionalities, where applicable

Figura 2.5: Parámetros del Comando DX

- **IS (Queried Sampling):** Este comando realiza solicita al dispositivo especificado, lo que se conoce como un 'IO Sample', que no es más que una muestra única de los valores de cada canal analógico o digital de los pines configurados en el dispositivo.

- **IR (Periodic Sampling):** De forma similar al comando anterior, este comando realiza una lectura periódica de los valores de cada canal dado un intervalo.

  Es el parámetro comprendido entre 0 y 0xFFFF quien define esa cantidad de tiempo en milisegundos.

- **IC (Change Detection Sampling):** De forma alternativa, es posible configurar los dispositivos para que envíen su estado cada vez que haya un cambio en su canal de entrada. El parámetro asociado, especifica una máscara de bits en los que cada bit activado representa el pin a monitorizar.

- **ND (Network Discovery):** Este comando obliga a todos los dispositivos de la red a identificarse. Entre la información devuelta se puede encontrar datos como la dirección larga, nombre y tipo de dispositivo.

2.4 KiCad

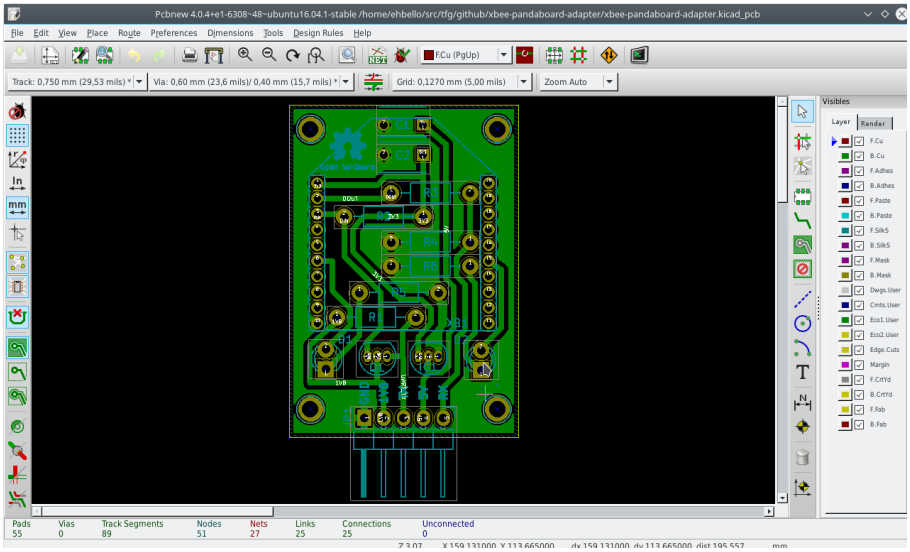


Figura 2.6: Ventana de edición de KiCad

KiCad es un entorno de software libre usado para el diseño de circuitos eléctricos, muy flexible y adaptable, en el que se pueden crear y editar un gran número de componentes y usarlos en Eeschema. KiCad permite el

diseño de circuitos impresos modernos de forma sencilla e intuitiva. Además, en Pcbnew, los circuitos se pueden diseñar con múltiples capas y ser visualizados en 3D.

2.5 Snappy Ubuntu Core 16

De entre todas las distribuciones que en su momento dieron soporte a la placa de desarrollo PandaBoard, surge **Snappy Ubuntu Core**, un sabor de Ubuntu destinada a dispositivos embebidos, entre otras cosas por su ligereza sin renunciar a su potencia.

Esta distribución destaca por proporcionar actualizaciones transaccionales con un riguroso aislamiento de aplicaciones y posibilidad de rollback. Esto garantiza la integridad de todos aquellos dispositivos que deban funcionar de forma autónoma, como es el caso.

Además, esta distribución se caracteriza por incluir el nuevo sistema de empaquetado de aplicaciones 'Snapd', el cual facilita a los desarrolladores la construcción y el mantenimiento de las aplicaciones para este sistema.

Este sistema de empaquetado está siendo ampliamente aceptado por el resto de distribuciones, lo que significa que nuestro trabajo no quedaría obsoleto si fuera necesario cambiar de distribución.

Snappy Ubuntu Core, en su versión 16, a diferencia de su distribución padre, Ubuntu Desktop, está en constante evolución y eso se nota a la hora de seguir sus pasos.

Respecto a su predecesora, la versión 15.04 incorpora grandes cambios tanto en la creación y gestión de paquetes snap, sustituyendo el antiguo

snappy por *snapd*, así como en la creación de imágenes con la herramienta *ubuntu-device-flash,* que aún no está completamente definida más allá de las arquitecturas comunes.

Aún así, se ha elegido la última versión de esta distribución, a modo de reto y con el objetivo de que las modificaciones realizadas perduren en el tiempo y sean de utilidad para el resto de la comunidad.

2.5.1 GNU/Linux kernel

El **kernel o núcleo de Linux**, en su arquitectura 'armhf' se puede definir como el corazón de este sistema operativo, encargado de que el software y el hardware de nuestra placa puedan trabajar juntos.

Sin duda se merece un apartado propio en esta memoria pues juega un papel importante en el soporte de la placa PandaBoard.

2.5.2 U-Boot

Toda distribución de Linux necesita un bootloader o cargador de arranque encargado de iniciar el resto del sistema operativo según los parámetros de configuración necesarios.

Es **U-Boot**, también conocido como *Universal Bootloader* el elegido para realizar esta función en la arquitectura 'armhf' debido a su gran flexibilidad y su desarrollo más activo.

2.5.3 Snapcraft 2.x

Snapcraft es la herramienta encargada de crear los paquetes *snap*, contenedores de la aplicación a instalar. Está ampliamente soportada por distribuciones popularmente conocidas como Ubuntu, ArchLinux, debian,

gentoo linux, fedora, openSUSE, openembedded, yocto project y OpenWRT.

Crear un paquete *snap* es tan sencillo como rellenar un fichero de sintaxis YAML y ejecutar la orden **snapcraft** encargada de descargar, construir y compactar los ficheros necesarios para instalar la aplicación.

Tal y como describe la documentación de la herramienta, un sencillo ejemplo podría ser el siguiente:

```
name: hello
version: "2.10"
summary: GNU Hello, the "hello world" snap
description: GNU hello prints a friendly greeting.
  This is part of the snapcraft tour at
https://snapcraft.io/create/
confinement: strict

apps:
  hello:
    command: hello

parts:
  gnu-hello:
    plugin: autotools
    source: http://ftp.gnu.org/gnu/hello/hello-2.10.tar.gz
```

Figura 2.7: Ejemplo de paquete snap básico

La versión 2.15.1 es la que actualmente incorpora la última versión Snappy Ubuntu Core. Tal y como se comentó en dicho apartado, la herramienta está en constante evolución y aporta significantes mejoras respecto a la rama anterior 1.x., incluyendo más plugins y automatizando acciones que en dicha versión hubieran debido realizarse de forma manual.

2.6 Domoticz

Domoticz es un sistema domótico libre que permite supervisar y configurar dispositivos como: Luces, interruptores, sensores/medidores de temperatura, lluvia, viento, UV, electricidad, gas, etc.

Diseñado para trabajar en varios sistemas operativos, su base está escrita en el lenguaje C++ y su interfaz de usuario es un front-end web escalable en HTML5 que se adapta automáticamente a ordenadores o dispositivos móviles.

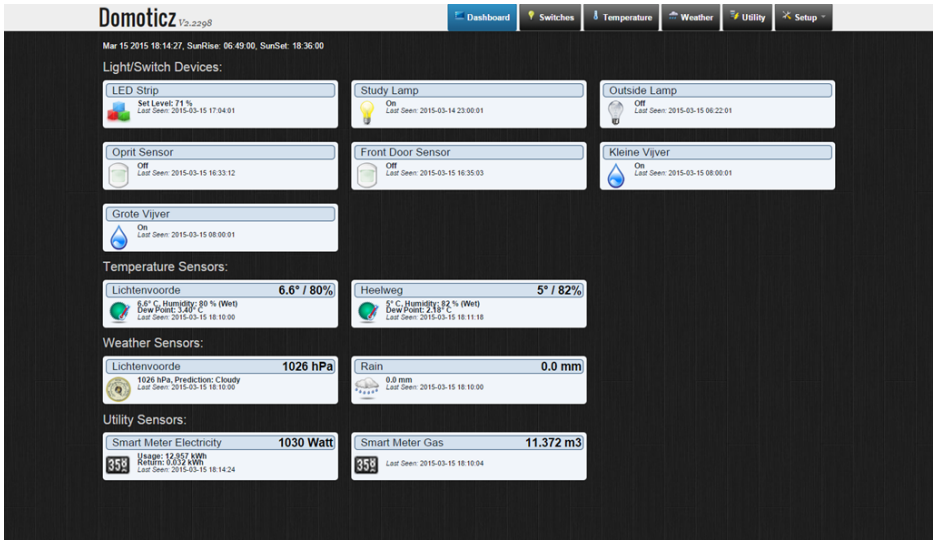


Figura 2.8: Vista del Dashboard de Domoticz

Además proporciona una API JSON que permite el control externo del sistema y facilita que por ejemplo, existan aplicaciones para teléfonos móviles Android que sirvan como control remoto.

A pesar de que no soporta dispositivos XBee, soporta una amplia variedad de otros dispositivos a través de métodos de conexión de tipo *serial*, *ethernet* y *mqtt*, lo que es capaz de aportar un añadido extra a nuestro proyecto.

Es por ello que este sistema el candidato ideal que aporte la experiencia en la gestión domótica.

2.7 Protocolo MQTT

El protocolo MQTT (MQ Telemetry Transport) o también conocido como **mosquitto**, es un protocolo de conectividad máquina a máquina extremadamente ligero diseñado sobre el protocolo TCP/IP para el transporte de mensajes basados en el patrón suscripción/publicación.

Es útil para conexiones remotas donde es necesario un footprint de código pequeño para reducir el ancho de banda.

Para ello es necesario un broker encargado de mantener y distribuir los mensajes a los clientes interesados basándose en un mensaje *topic*.

Cada mensaje está compuesto por dicho *topic*, que no es más que una consecución de términos o niveles jerarquizados delimitados por una separador y un *payload* que puede ser cualquier tipo de texto, por ejemplo un valor o una cadena json.

Figura 2.9: ejemplo de topic MQTT

Cualquier cliente tendrá la opción de suscribirse a uno o más *topics* y será capaz de recibir los mensajes asociados, descartando el resto de mensajes no relacionados con el *topic* suscrito.

Existe, además, la posibilidad de hacer uso de los siguientes comodines:

- Single Level (+): como el nombre sugiere, un comodín de nivel simple es sustituido por un nivel de *topic*.

Figura 2.10: ejemplo de comodín de nivel simple

- Multi Level (#): mientras el comodín de nivel simple sólo cubre un nivel, el comodín multinivel cubre un número arbitrario de niveles de *topic*.

Figura 2.11: ejemplo de comodín de nivel múltiple

2.8 xbee2mqtt

xbee2mqtt es un demonio de licencia GPLv3 desarrollado en Python, capaz de monitorizar los mensajes entrantes de un XBee conectado a un puerto de serie de nuestro ordenador. Cada mensaje recibido es distribuido a un broker MQTT cumpliendo con el formato de este protocolo.

Entre otras, sus funcionalidades son las siguientes:

- Procesa paquetes *Zigbee received packet (0x90)* y *ZigBee IO data sample (0x92)* procedentes del puerto de serie conectado al módulo XBee.
- Envía los valores recibidos de cada puerto digital (dio) o analógico (adc) a través del MQTT broker.

- Se basa en *mapeos* puerto:topic definidos en su fichero de configuración o en un patrón por defecto cuando no concuerdan con las direcciones y puertos asociados.
- Para los *topics* definidos asociados a algún determinado puerto, realiza la suscripción {topic} + "/set" que permite modificar el estado de los puertos digitales. En caso de no ser digital, fuerza la configuración.
- Además es capaz de preprocesar los valores recibidos antes de publicarlos en el broker.

2.9 Node-RED

Node-RED es una herramienta para relacionar de forma conjunta dispositivos hardware, APIs y servicios online.

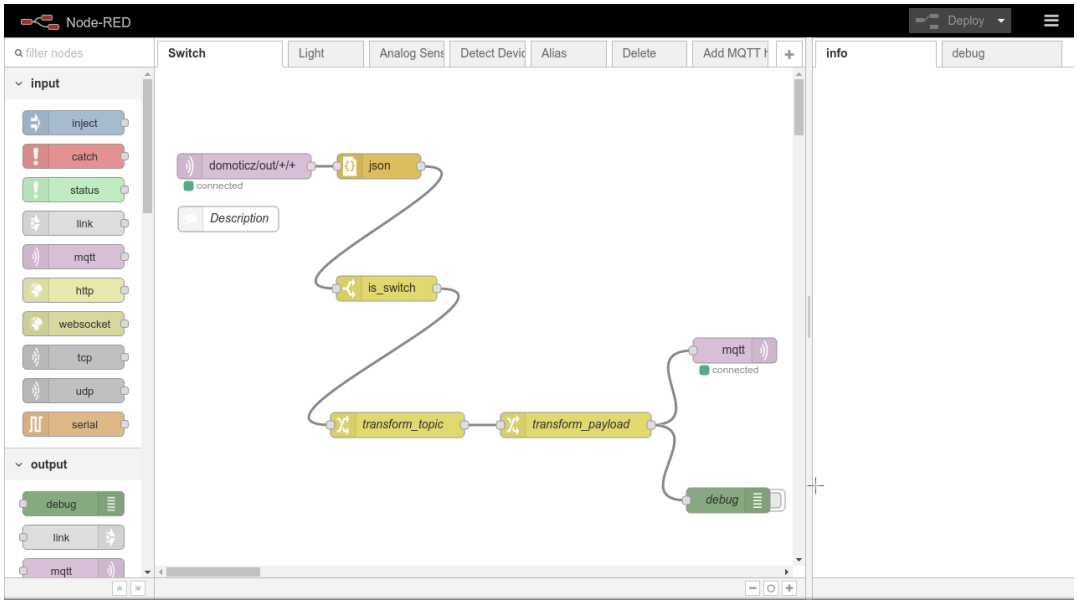


Figura 2.12: Vista del principal de Node-RED

Provee un editor web de flujos que facilita la relación a través de flujos usando un amplio rango de nodos disponibles en la paleta. Estos flujos pueden ser desplegados con un simple click.

Además, se pueden crear funciones JavaScript utilizando un editor de texto enriquecido. La librería integrada permite guardar las funciones más útiles, plantillas u otro flujos para su reutilización.

Está desarrollado sobre Node.js obteniendo las ventajas de su modelo no bloqueante conducido por eventos y especialmente diseñado para ejecutarse sobre hardware de bajo coste.

Cada flujo se guarda en formato JSON por lo que pueden ser fácilmente importados y exportados para su compartición.

2.10 QMLScenes

QMLScene es una utilidad que carga y muestra documentos QML pertenecientes a Qt5 que permite testear aplicaciones QML antes de que estén completadas.

QML es un lenguaje declarativo en formato JSON que permite describir interfaces de usuario basándose en sus componentes visuales y cómo interactúan entre ellos.

Usando el módulo QtQuick se pueden desarrollar interfaces de usuario de forma rápida sin necesidad de gestionar un desarrollo íntegro.

Capítulo 3
Trabajo realizado

Cumpliendo con los objetivos del proyecto, este apartado incluye una descripción detallada del trabajo realizado en cada componente.

3.1 Adaptador XBee a PandaBoard

Para comunicarse con un módulo XBee de forma serial, basta tener un puerto de serie al que conectar las señales de recepción (Rx), envío (Tx), corriente (3V3) y tierra (GND).

Sin embargo, conectarlo a una placa PandaBoard, no es tan sencillo. A pesar de que dispone tres de las cuatro patillas necesarias, los niveles LVTTL de ésta trabajan a voltajes diferentes.

Es por ello que surge la necesidad de diseñar un adaptador intermedio que se encargue de realizar las conversiones necesarias.

3.1.1 Puertos de expansión en PandaBoard

La PandaBoard dispone de dos puertos de expansión (J3 y J6) de 28 pines cada uno, ubicados en uno de los extremos de la placa.

Entre ellos podemos encontrar pines con funciones GPIO, I2C, USB, SPI o, las que nos interesan: UART.

De las tablas de definición de pines extraídas del manual de referencia e incluidas en el apéndice, podemos destacar los siguientes pines del EXPANSION CONNECTOR A

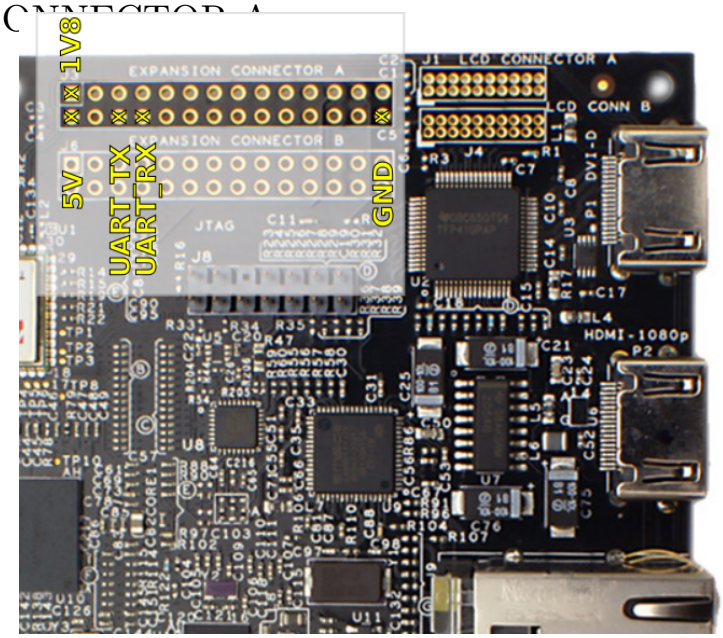


Figura 3.1: Descripción de pines necesarios para conectar el adaptador XBee

3.1.2 Diseño de la PCB

Se ha comentado en la tabla 2.1 del capítulo previo que, una de las características de los módulos XBee Series2 es que trabajan con una corriente de entre 2.8 y 3.6V.

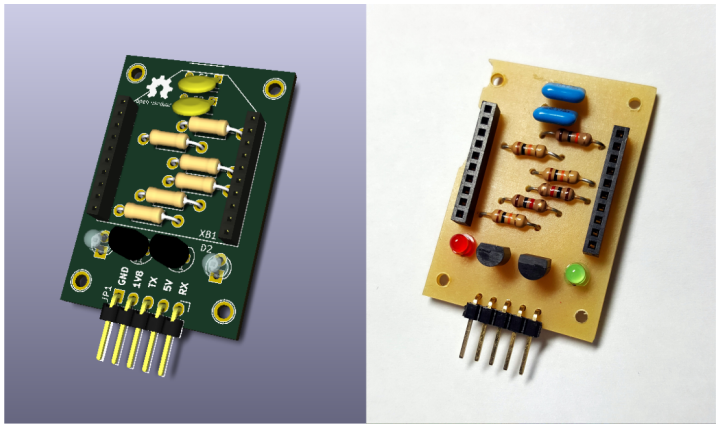


Figura 3.2: PCB 3D vs PCB DiY (frente)

Sin embargo, en ninguno de los pines de la PandaBoard proporciona este

nivel de corriente de forma directa. Es por ello que se ha usado el pin de 5V para bajar la corriente a 3.3 mediante un divisor de resistencias.

Por otro lado, según las especificaciones del procesador OMAP4460, este trabaja con corrientes de bajo voltaje. En este caso, 1.8V. Para convertir entonces los niveles LVTTL de corriente entre ambas placas, se ha usado un regulador de voltaje MCP3307 y un transistor MOSFET común 2N7000.

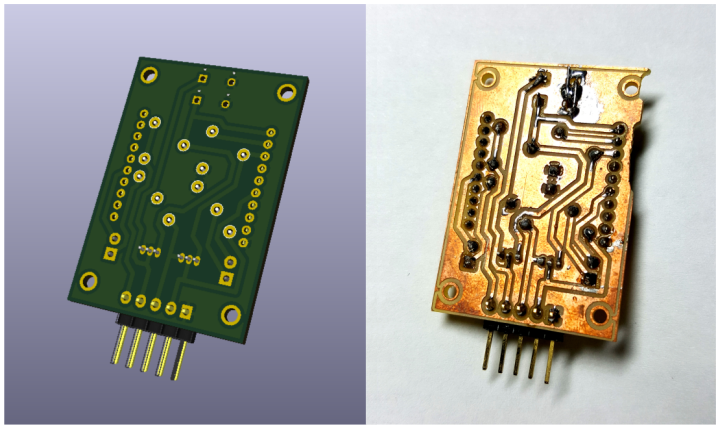


Figura 3.3: PCB 3D vs PCB DiY (reverso)

Finalmente y, como se puede ver en las tablas 3.2 y 3.3, se ha usado un encapsulado THT (Tecnología de agujeros pasantes) para que sea más fácil de fabricar de forma casera si fuera necesario.

3.2 Distribución Linux

Como se ha comentado en el capítulo anterior de Conocimientos Previos, el estado actual de la placa PandaBoard es de casi absoluta obsolescencia por parte del fabricante y a duras penas recibe soporte por parte de la comunidad.

Esto hace que las versiones más actuales de las distribuciones comunes puedan no funcionar *out-of-the-box.*

No obstante, dadas las ventajas de la última versión de Ubuntu Core y para dar continuidad a nuestro trabajo, se ha tomado como un reto interesante la adaptación del soporte previo a esta versión.

3.2.1 Imagen distribuíble

Partimos desde el final: la creación de un fichero imagen de la tarjeta de memoria, que facilite la descarga e instalación de la distribución en nuestra PandaBoard con un simple volcado.

Para ello, Ubuntu dispone de una herramienta de nombre **ubuntu-device-flash**.

Esta herramienta crea una imagen de un mínimo de 4GB partiendo de tres tipos de snap distintos:

- De **tipo OS**. Concretamente el *ubuntu-core_*.snap):*, que contiene el los ficheros relacionados con el Sistema Operativo.
- De **tipo Kernel**. que, como su nombre indica, es el encargado de instalar el núcleo de Linux en la partición de arranque correspondiente.
- De **tipo Gadget**. El más místico de todos, pues está siendo redefinido según evoluciona el desarrollo. Encargado de instalar el bootloader elegido en el soporte de almacenamiento, definir la arquitectura, la estructura de particiones y los paquetes preinstalados en la imagen.

Si estos snap's no son proporcionados de forma local, esta herramienta se encarga de descargarlos por sí sola desde la Ubuntu Store de forma

automática.

Durante el desarrollo de este proyecto se han usado diferentes versiones de esta herramienta publicadas de forma no oficial en el espacio personal de Michael Voght, el encargado de desarrollarla.

Actualmente es posible instalar una versión capaz de generar imágenes para Ubuntu Core 16, ejecutando:

```
$     snap install ubuntu-device-flash
```

Sin embargo, la versión actual no permite la instalación de snap's de forma local, sin pasar por la Ubuntu Store. No obstante, es esperable que un futuro próximo sí lo haga.

Uno de los puntos claves de este apartado ha sido localizar la versión funcional más adecuada para la generación de la imagen distribuíble.

3.2.2 Bootloader

En la estructura de los snap de tipo Gadget, los parámetros que definen la instalación del bootloader son los siguientes:

```
boot-assets:
    files:
      - path: uEnv.txt
      - path: MLO
      - path: u-boot.img
      - path: uboot.env
    raw-files:
      - path: MLO
        offset: 131072 # @ 128kB
      - path: u-boot.img
        offset: 393216 # @ 384kB
```

Figura 3.4: Parámetros de configuración del bootloader en un snap

No solo se encuentra el binario del bootloader (u-boot.img), sino también ficheros como los que contienen las variables de entorno: uEnv.txt y uboot.env.

Estas variables de entorno están compuestas por otras variables o algunas de las funciones proporcionadas por el bootloader. Todas ellas, escritas de forma lógica, se encargan de conseguir el arranque de la distribución, localizando y cargando elementos como el dtb (Device Tree Binaries), el initrd o el kernel.

El primero de los ficheros *uEnv.txt* es un fichero de texto plano, que permite al usuario definir algunas de las variables que desee solapar.

El otro de los ficheros *uboot.env* es un fichero binario que, en su lugar, contiene absolutamente todas las variables de entorno U-Boot a cargar en memoria.

En cualquier momento del arranque, alguna de las variables (por no llamarlas funciones) puede actualizar el contenido de este entorno con las órdenes *setenv* y *saveenv.*

La herramienta *ubuntu-device-flash* pone como requisito proporcionar este fichero *uboot.env* para asegurarse de que se carga un entorno completo y válido. Su objetivo es insertar en tiempo de generación, las variables necesarias que definen el kernel instalado.

Si este fichero no se proporciona, el uboot cargaría un entorno por defecto definido en su binario.

Este fichero se puede generar un fichero plano con la orden:

```
$     mkenvimage -r -s 131072 -o uboot.env uboot.env.in
```

El binario resultante, tiene un checksum CRC que el bootloader comprobará durante el arranque. Si por alguna razón no se genera correctamente, el mensaje devuelto durante el arranque será:

```
uboot.env: *** Warning - bad CRC, using default environment
```

En la actualidad, esta intromisión por parte de la herramienta, hace que el fichero resultante sea generado de forma incorrecta causando este error.

Es por ello que, a modo de *workaround* temporal, se ha incluido cierto contenido en el fichero *uEnv.txt*. De otra forma, hubiera tenido que estar vació.

Otra forma de solucionarlo, hubiera sido ejecutando los dos siguientes comandos en la partición de arranque de la imagen:

$ strings uboot.env > uboot.env.in

$ mkenvimage -s 131072 -o uboot.env uboot.env.in

3.2.3 Kernel

En la Ubuntu Store, existe un kernel genérico para la arquitectura *armhf* de nombre linux-arm basado en el set de configuraciones por defecto *multi_v7_defconfig* del kernel.

No obstante, este set de configuraciones no es suficiente para arrancar con éxito el hardware de la PandaBoard, pues carece de flags como el CONFIG_SQUASHFS=y necesario para poder leer el contenido del fichero y seguir cargando el initrd, resultando en el siguiente error:

```
initramfs: findfs: unable to resolve 'LABEL=writable'
```

Entonces ha sido necesario crear un snap que compila la rama estable del kernel de linux, actualmente la 4.4 en su última versión 4.4.19, eligiendo el set de configuraciones por defecto definido por **omap2plus_defconfig** y activando los siguientes configuraciones en el kernel:

- Activación del sistema de ficheros SQUASFS

  CONFIG_SQUASHFS=y

  CONFIG_SQUASHFS_XZ=y
- Activación de pseudoterminales devpts

  CONFIG_DEVPTS_MULTIPLE_INSTANCES=y
- Activación de framebuffer para carga de logos durante el arranque

  CONFIG_FB_PRE_INIT_FB=y

  CONFIG_FB_OMAP2=y
- Activación del Real Time Clock

  CONFIG_RTC_DRV_TWL4030=y
- Activación de USBs y Periféricos USB

  CONFIG_USB=y

  CONFIG_USB_KBD=y

  CONFIG_USB_MOUSE=y

  CONFIG_USB_OHCI_HCD=y

  CONFIG_USB_EHCI_HCD=y

  CONFIG_USB_STORAGE=y
- Activación del subsistema de vídeo

  CONFIG_OMAP2_DSS=y

CONFIG_DISPLAY_CONNECTOR_DVI=y

CONFIG_DISPLAY_CONNECTOR_HDMI=y

CONFIG_DISPLAY_PANEL_DPI=y

Además, se incluyen los binarios de firmware para los módulos inalámbricos WL1xxx.

3.3 Interfaz de usuario y control remoto

3.3.1 Análisis y decisiones tomadas

Antes de tomar la decisión de implementar una interfaz de control desde cero, se realizó un análisis de las interfaces de control domótico existentes.

Se llegó a la conclusión de que hay interfaces de usuario de licencia libre bastante completas como OpenHAB o Domoticz y, si el objetivo era tener un producto lo más completo posible, desarrollar esta interfaz desde cero no sería la solución.

Es por ello que se decidió no reinventar la rueda y probar con OpenHAB, en sus versiones 1 y 2, desarrollada en Java y con una gran comunidad de usuarios detrás de ella. Sin embargo, después de un tiempo de prueba, la conclusión fue que sus grandes requisitos de rendimiento no se adaptaban para nada a un sistema embebido.

Dicho esto, la alternativa más viable y completa fue Domoticz. Con un aspecto visual atractivo, desarrollado en C++ y además disponía de una aplicación para móviles Android bastante completa.

3.3.2 Límites del proyecto

Para que el proyecto no se hiciera demasiado extenso, se asumió desde el primer momento que los módulos Xbee estaban preconfigurados en modo API y asociados entre sí por medio de la misma red.

Esto nos reduce el problema a implementar las acciones de creación, eliminación, configuración, lectura y escritura de sensores.

En el repositorio de código del proyecto se proporcionan los perfiles de configuración para XBee's en modo *Coordinador* y modo *nodo final*.

3.3.3 Adaptación de Domoticz

La idea inicial y lógica, parte de implementar una nueva interfaz hardware que permitiera a Domoticz entender cómo funcionan los módulos XBee.

No obstante, a pesar de lo que pudiera parecer por su aspecto visual y la cantidad de funcionalidades que proporciona, la calidad del código de desarrollo deja mucho que desear. No era posible implementar algo funcional en tiempo limitado.

Aprovechando la base de conocimiento adquirida en esta interfaz, la alternativa fue aprovechar su soporte para el protocolo MQTT y realizar la comunicación con el coordinador XBee por medio de este.

Entre las funcionalidades implementadas podemos destacar las siguientes:

- Se ha activado la opción de crear dispositivos MQTT directamente desde el hardware MQTT Client de la sección de hardware.
- A cada dispositivo creado, se le asocia un DeviceTopic de tipo

<address>/<port> que permite identificar la dirección y el puerto al que está asociado.

- Se han añadido la posibilidad de crear dispositivos si se reciben los payloads adecuados por el protocolo MQTT, lo que permite que el Node Discovery sea posible y cree dispositivos en consecuencia.
- Ahora se envía información al broker MQTT sobre los eventos de eliminación y creación de dispositivos en Domoticz para que sean correctamente interpretados.

3.3.4 Adaptación de xbee2mqtt

En primer lugar, el código obtenido del repositorio del autor original no funcionaba correctamente. Las acciones iniciales partían de las modificaciones necesarias en el código para recuperar el funcionamiento de la aplicación.

Además, para cubrir todas las funcionalidades descritas en los límites del proyecto, esta pasarela de XBee a MQTT debía realizar ciertas acciones con los módulox XBee que aún no estaban implementadas.

Algunas de estas funcionalidades implementadas fueron:

- Soporte para la librería python-xbee oficial y en consecuencia soporte para los módulos XBee Series2
- Interpretación de los paquetes relacionados con el Node Discovery y Node Identification.
- Manejo de las respuestas a comandos causantes de alguna petición.
- Suscripción a topics siguiendo el patrón por defecto, lo que permite la

gestión dinámica de los XBee's de la red sin necesidad de añadirlos en el fichero de configuración.

- Manejo de la configuración de pines.
- Configuración del IO Change Detection y el IO Sample Rate basándose en los parámetros de configuración correspondientes.

3.3.5 Definición de flujos Node-RED

Como alternativa a la interfaz hardware en Domoticz, se consiguió que tanto esta interfaz como la pasarela xbee2mqtt fueran capaces de realizar las funciones necesarias.

No obstante, pesar de hablar el protocolo MQTT, no eran capaces de entenderse entre sí pues sus mensajes no tenían la misma estructura.

Por un lado Domoticz enviaba y recibía mensajes en formato JSON mediante los topics domoticz/out y domoticz/in respectivamente y, or otro lado, la pasarela xbee2mqtt enviaba y recibía todos sus mensajes en forma numérica a través de los topics /raw/xbee/out/ y /raw/xbee/in

Era necesario añadir un nuevo componente que tradujera mensajes de uno a otro, recibiendo mensajes de uno y republicando su mensaje convertido.

Se podía haber implementado un nuevo programa en python que realizara la función de manera directa. Sin embargo, parecía más interesante el uso de Node-RED, especiamente diseñado para crear y ampliar estos flujos de forma gráfica e intuitiva.

Además, esta funcionalidad no sólo podría servir para la comunicación

XBee, sino para otro tipo de necesidades que tuviera el usuario.

3.4 Modo kiosko para salida de vídeo

Finalmente, y cumpliendo con el último requisito, se ha creado un paquete snap independiente que, haciendo uso del lenguaje QML, ellos módulos QtQuick y QtWebkit, y la herramienta QMLScenes es capaz de generar una ventana simple en pantalla completa que cargue y navegue la dirección web local http://localhost. Todo ello sobre un servidor gráfico Mir.

```
import QtQuick 2.1
import QtWebKit 3.0

WebView {
   width: 1024
   height: 750
   url: "http://localhost/"
}
```

Figura 3.5: Definición de vista Web en ventana usando QML

3.5 Empaquetado

Para facilitar la instalación, mantenimiento y actualización de todo el trabajo realizado, se ha empaquetado en forma de paquete *snap* haciendo uso de la herramienta **snapcraft**.

Sobre todo para la generación del paquete linux-panda.snap, ha sido necesaria la realización de ciertas modificaciones y mejoras sobre el código de la versión 2.15.1 en las que se se reducen ciertos tiempos de descarga y se activa la posibilidad de compilación cruzada para ciertos plugins.

Capítulo 4
Puesta en funcionamiento

Instalación de las herramientas necesarias

```
$     apt-get install snapcraft
$     wget https://people.canonical.com/~mvo/all-snaps/obsolete/ubuntu-device-flash
```

Construcción del Gadget:

```
$     cd armhf-snaps
$     make MACHINE=panda
```

Construcción del Kernel:

```
$     cd linux-armhf-snap
$     snapcraft login
$     sudo snapcraft -target-armhf
```

Instalación de herramientas necesarias en una máquina de arquitectura armhf para realizar la complilación.

```
$     sudo snap install classic --devmode --beta
$     sudo classic.create
$     sudo classic.shell
$     sudo apt update
$     sudo apt install snapcraft git-core
```

Descarga del repositorio:

```
$     git clone https://github.com/ULL-InformaticaIndustrial-
Empotrados/TFG_ZigBee_Embedded_Controller
```

Construcción de la pila de aplicaciones:

```
$     cd snap
$     snapcraft

$     cd mir-kios-browser-snap
$     snapcraft
```

Generación de la imagen:

```
$ sudo -E UBUNTU_DEVICE_FLASH_IGNORE_UNSTABLE_GADGET_DEFINITION=1 ./ubuntu-device-
flash --verbose core 16 --size=4 --enable-ssh --channel=edge --output=sna
ppy-panda.img --gadget panda_*_all.snap --kernel linux-panda_*_armhf.snap --os
ubuntu-core  --developer-mode
```

Volcado de la imagen a la tarjeta de memoria:

```
sudo dd if=snappy-panda.img bs=4k of=/dev/sdX
```

Capítulo 5
Conclusiones y líneas futuras

Para acabar, decir que nuestro proyecto, lo que ha pretendido es, proporcionar una solución económica y altamente personalizable. Ha supuesto un gran reto multidisciplinar enfocado a la integración de elementos de diferentes características para conseguir completar nuestro fin. El fin de que diferentes aparatos electrónicos, que hasta ahora han trabajado de forma individual, puedan cooperar entre sí de una forma completamente innovadora, gracias a Internet.

Algunas de las líneas futuras a seguir están relacionadas con la ampliación de las acciones realizadas por los distintos elementos sobre el protocolo MQTT, así como asegurar y mejorar el soporte los elementos instalados sobre el hardware.

Capítulo 6
Summary and Conclusions

Finally, to say that our project, which has sought is to provide an economical and highly customizable solution. It has been a great challenge multidisciplinary focused on the integration of elements of different features to achieve complete our end. So that different electronic devices, which so far have worked individually, can cooperate with each other in a completely innovative way, thanks to the Internet.

Some of the future lines to follow are related to the expansion of the actions taken by the various elements on the MQTT protocol and ensure and improve support elements installed on the hardware.

Capítulo 7
Presupuesto

A continuación se presenta un análisis estimado de los recursos materiales y humanos para la consecución de este proyecto.

7.1 Costes en recursos humanos

Concepto	**Horas**	**Precio/Hora (€)**	**Subtotal**
Desarrollo	420	10	4200
Puesta en marcha	20	10	200
TOTAL			4400

Tabla 7.1: Resumen de costes en recursos humanos

7.2 Costes en recursos materiales

Concepto	Cantidad	Precio Unitario	Subtotal
PandaBoard ES (OMAP4460)	1	169,84	169,84
PowerSupply (5V, 5Amp)	1	16,9	16,9
PandaBoard Metal Case	1	32,0	32,0
MicroSDHC 16Gb Class10 + SD Adapter	1	6,95	6,95
XBee RF Module	1	22,0	22,0
2Layer Green PCB 5cm x 5cm Max (ITEAD Studio)	1/5	13,44	2,69
1µF Capacitor (Polarized)	2	0,17	0,34
LED 3mm	2	0,13	0,26
Voltage Regulator (MCP1700-3302E/TO)	1	0,44	0,44
Pin Header Male	1x5	0,37	0,37
MOSFET Transistor (2N7000)	1	0,37	0,37
10KΩ Resistor	4	0,32	1,28
1Ω Resistor	2	0,32	0,64
Jumper Wire	5	0,16	0,80
TOTAL			254,90

Tabla 7.2: Resumen de costes en recursos humanos

Capítulo 8
Apéndices

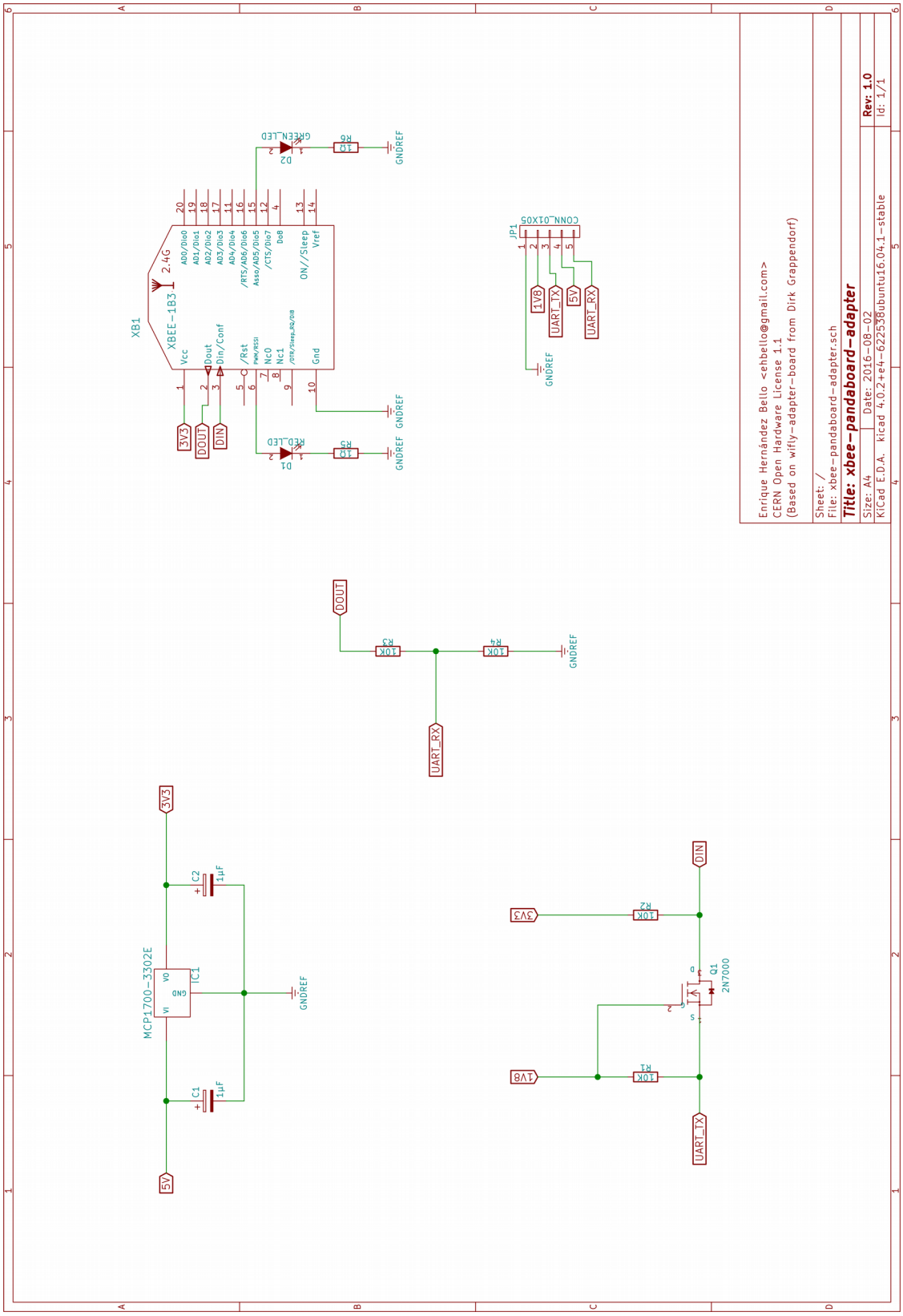
8.1 Definición de Pines del Conector de Expansión "A" en la PandaBoard

J3 Pin #	OMAP Ball #	Primary Function	Secondary Function	Description of Pandaboard ES Usage
1	---	VIO_1V8	---	1.8V I/O Power
2	---	DCIN_JACK	---	5Vdc Input Power
3	B16	GPMC_AD7	SDMMC2_DAT7	GPMC Address/Data Bit 7
4	AH23	MCSPI1_CS3	GPIO_140	SPI1 Chip Select 3 (also UART1_RTS)
5	A16	GPMC_AD6	SDMMC2_DAT6	GPMC Address/Data Bit 6
6	AH19	UART4_TX	GPIO_156	UART4 Transmit Data
7	D15	GPMC_AD5	SDMMC2_DAT5	GPMC Address/Data Bit 5
8	AG20	UART4_RX	GPIO_155	UART4 Receive Data
9	C15	GPMC_AD4	SDMMC2_DAT4	GPMC Address/Data Bit 4
10	AF23	MCSPI1_CS1	GPIO_138	SPI1 Chip Select 1 (also UART1_RX)
11	D13	GPMC_AD3	SDMMC2_DAT3	GPMC Address/Data Bit 3
12	AG22	MCSPI1_SIMO	GPIO_136	SPI1 Slave In Master Out
13	C13	GPMC_AD2	SDMMC2_DAT2	GPMC Address/Data Bit 2
14	AG23	MCSPI1_CS2	GPIO_139	SPI1 Chip Select 2 (also UART1_CTS)
15	D12	GPMC_AD1	SDMMC2_DAT1	GPMC Address/Data Bit 1
16	AE23	MCSPI1_CS0	GPIO_137	SPI1 Chip Select 0
17	C12	GPMC_AD0	SDMMC2_DAT0	GPMC Address/Data Bit 0
18	AE22	MCSPI1_SOMI	GPIO_135	SPI1 Slave Out Master In
19	B12	GPMC_NWE	SDMMC2_CMD	GPMC Write Enable
20	AF22	MCSPI1_CLK	GPIO_134	SPI1 Clock Out
21	B11	GPMC_NOE	SDMMC2_CLK	GPMC Output Enable
22	D19	GPMC_AD15	GPIO_39	GPMC Address/Data Bit 15
23	AH22	I2C4_SDA	GPIO_133	I2C4 Serial Data
24	AG21	I2C4_SCL	GPIO_132	I2C4 Serial Clock
25	---	REGEN1	---	TWL6030 REGEN1
26	E7	SYS_NRESPWRON	---	Power On Reset
27	---	DGND	---	Digital Ground
28	---	DGND	---	Digital Ground

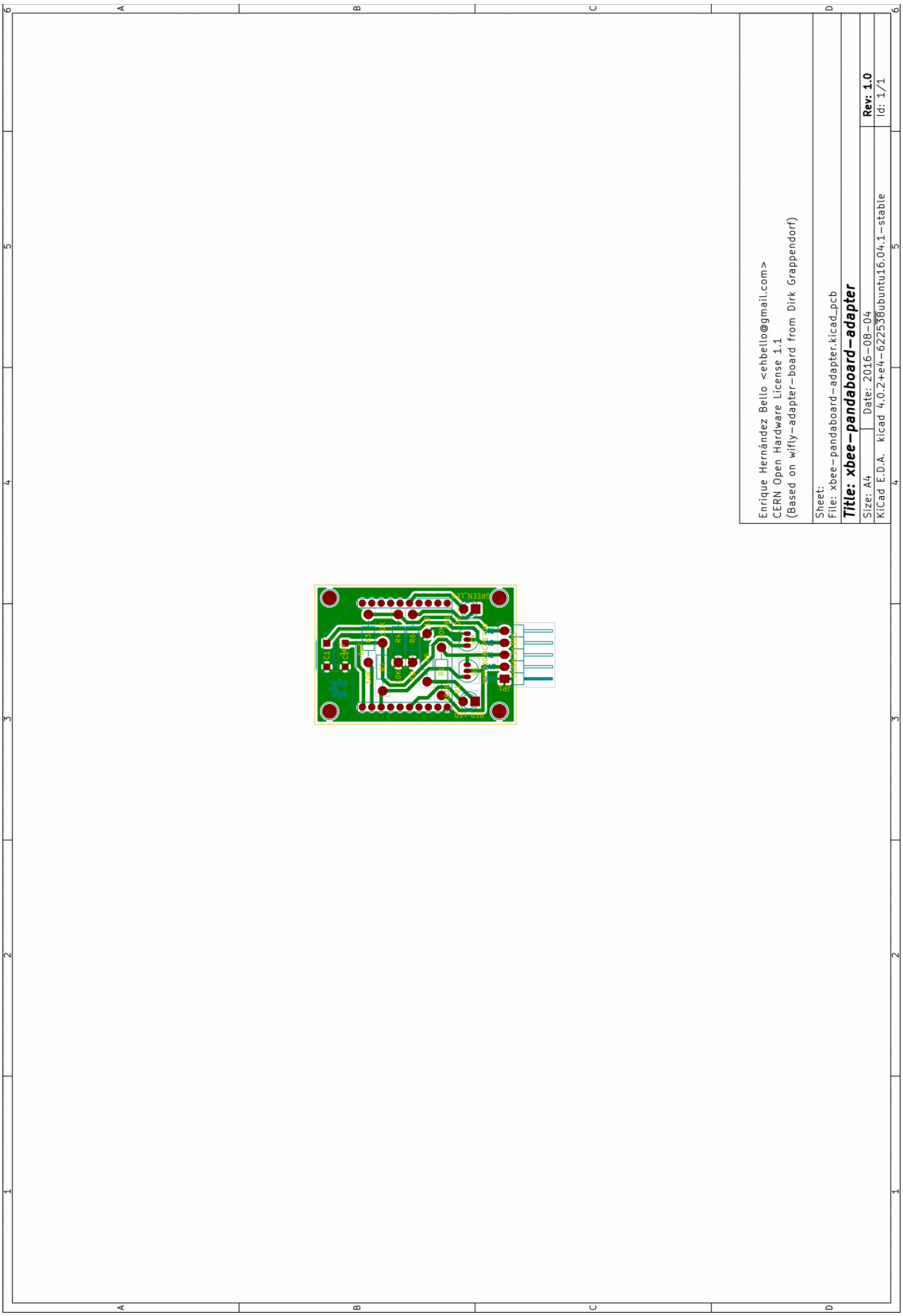
8.2 Definición de Pines del Conector de Expansión "B" en la PandaBoard

J6 Pin #	OMAP Ball #	Primary Function	Secondary Function	Description of Pandaboard ES Usage
1	---	VBUS_3	---	VBUS out from USB Host Port #3
2	---	VBUS_4	---	VBUS out from USB Host Port #4
3	---	USBH3_DM	---	USB Host Port #3 Data Minus
4	---	USBH4_DM	---	USB Host Port #4 Data Minus
5	---	USBH3_DP	---	USB Host Port #3 Data Plus
6	---	USBH4_DP	---	USB Host Port #4 Data Plus
7	---	DGND	---	Digital Ground
8	---	DGND	---	Digital Ground
9	C19	GPMC_AD14	GPIO_38	GPMC Address/Data Bit 14
10	D18	GPMC_AD13	GPIO_37	GPMC Address/Data Bit 13
11	AF7	SYS_NRESWARM	---	Warm Reset
12	---	PB_POWER_ON	---	Power on input to TWL6030 (ref. to VBAT)
13	---	HFL_P	---	Hands Free Left Speaker Out (+)
14	AG24	H_DMTIMER11_PWM	GPIO_121	Display PWM Control
15	---	HFL_N	---	Hands Free Left Speaker Out (-)
16	---	VDD_VAUX1	---	VAUX1 from TWL6030
17	C18	GPMC_AD12	GPIO_36	GPMC Address/Data Bit 13
18	C16	GPMC_AD8	GPIO_32	GPMC Address/Data Bit 8
19	B26	GPMC_WAIT0	GPIO_61	GPMC Wait input 0
20	D16	GPMC_AD9	GPIO_33	GPMC Address/Data Bit 9
21	C25	GPMC_NWP	GPIO_54	GPMC Write Protect
22	C17	GPMC_AD10	GPIO_34	GPMC Address/Data Bit 10
23	B22	GPMC_CLK	GPIO_55	GPMC Clock Out
24	D17	GPMC_AD11	GPIO_35	GPMC Address/Data Bit 11
25	B25	GPMC_NCS0	GPIO_50	GPMC Chip Select 0
26	D25	GPMC_NADV_ALE	GPIO_56	GPMC Address Valid/Address Latch Enable
27	C21	GPMC_NCS1	GPIO_51	GPMC Chip Select 1
28	C23	GPMC_NBE0_CLE	GPIO_59	GPMC Byte Enable 0/Command Latch Enable

8.3 Esquemas de diseño del adaptador



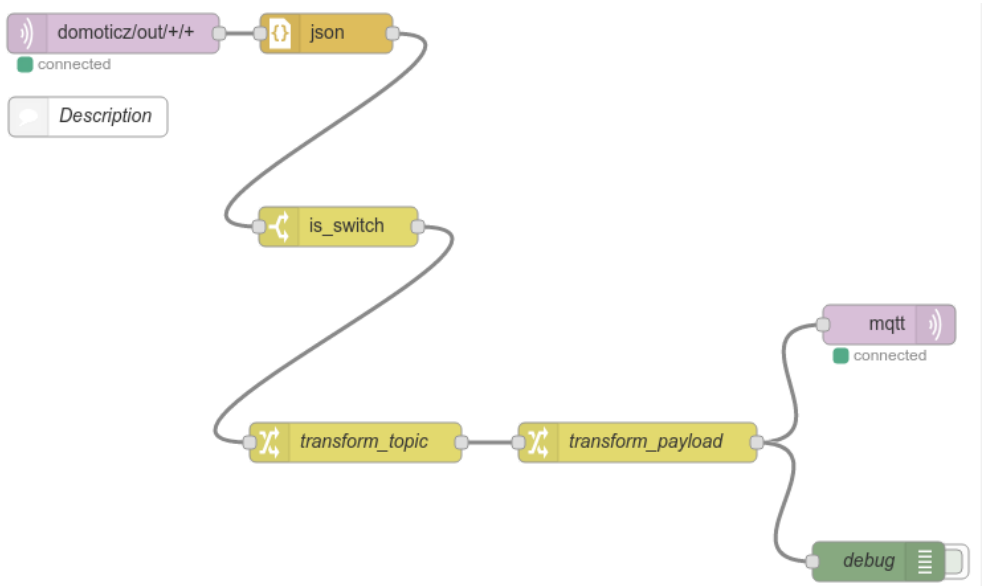
8.4 Layout PCB del adaptador



8.5 Flujos de Node-Red

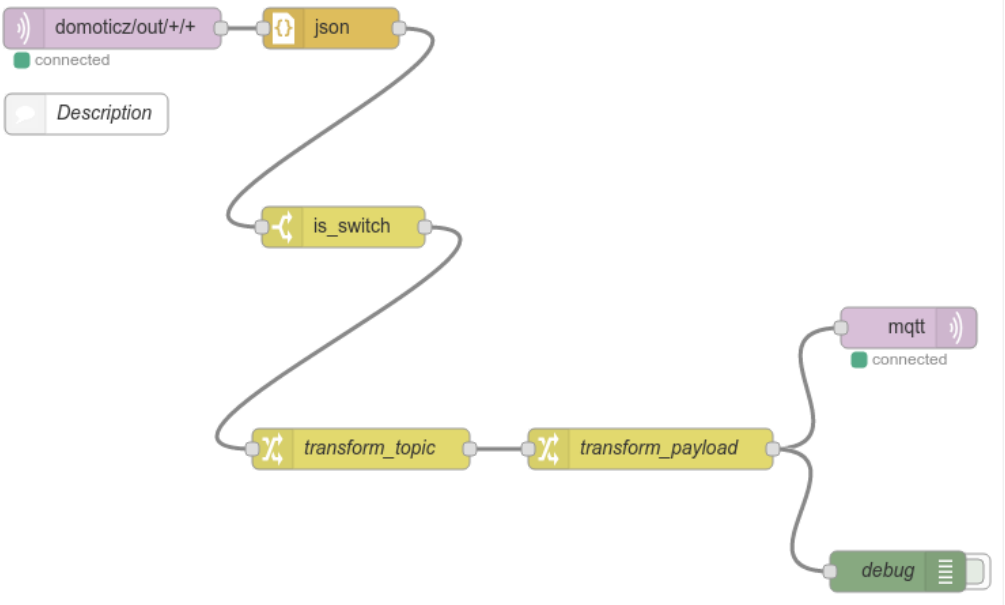
8.5.1 Switch

Conecta las



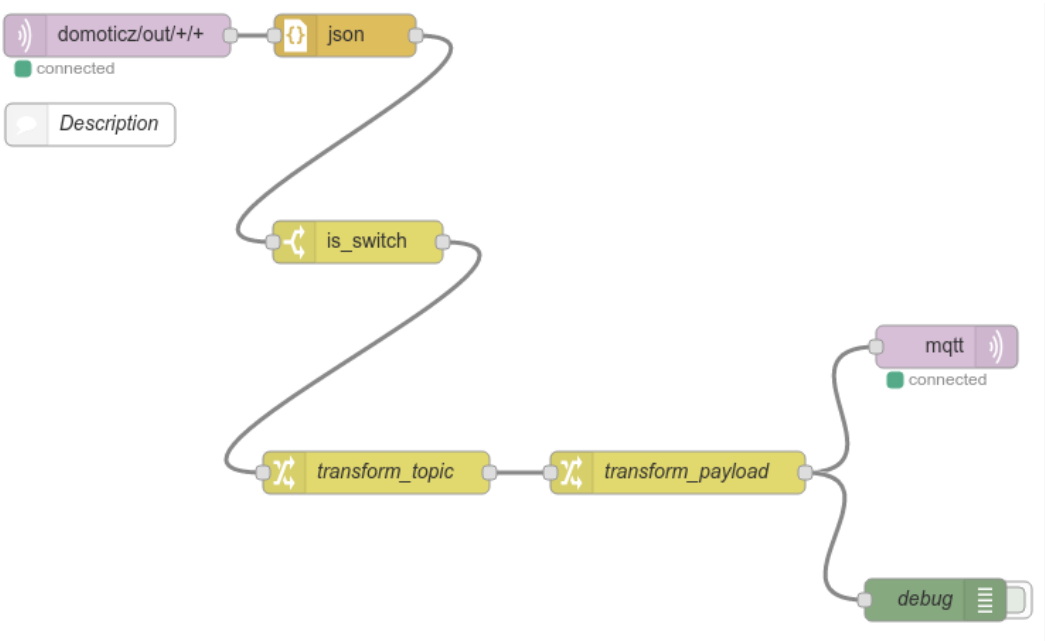
actualizaciones de los interruptores emitidas por Domoticz con un determinado pin del dispositivo XBee correspondiente.

8.5.2 Light



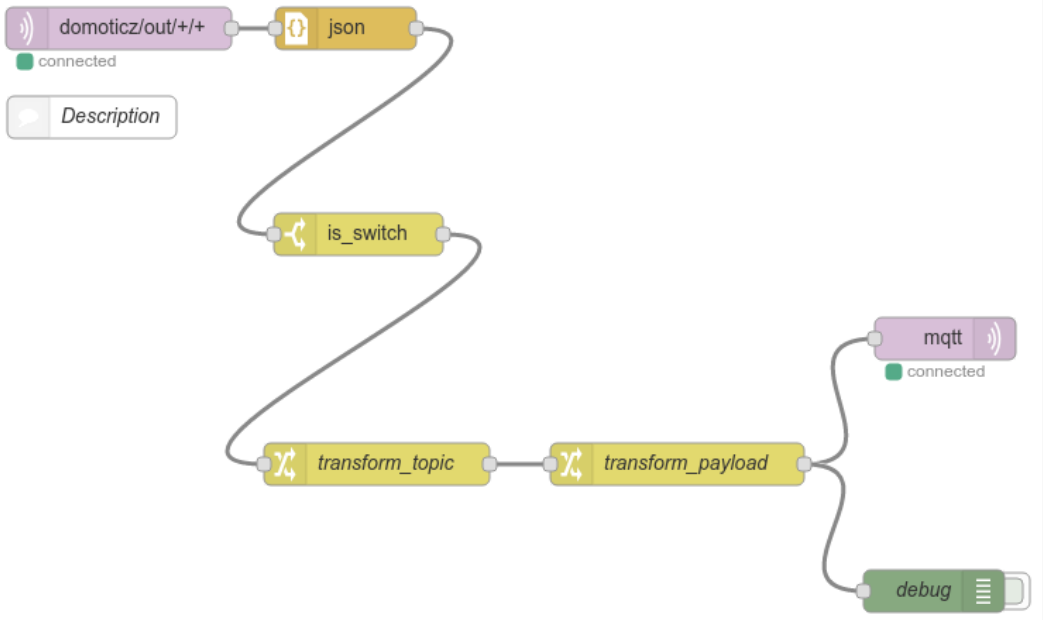
Conecta las actualizaciones de las entradas digitales emitidas por los dispositivos XBee con sensores digital determinados en Domoticz.

8.5.3 Analog Sensor



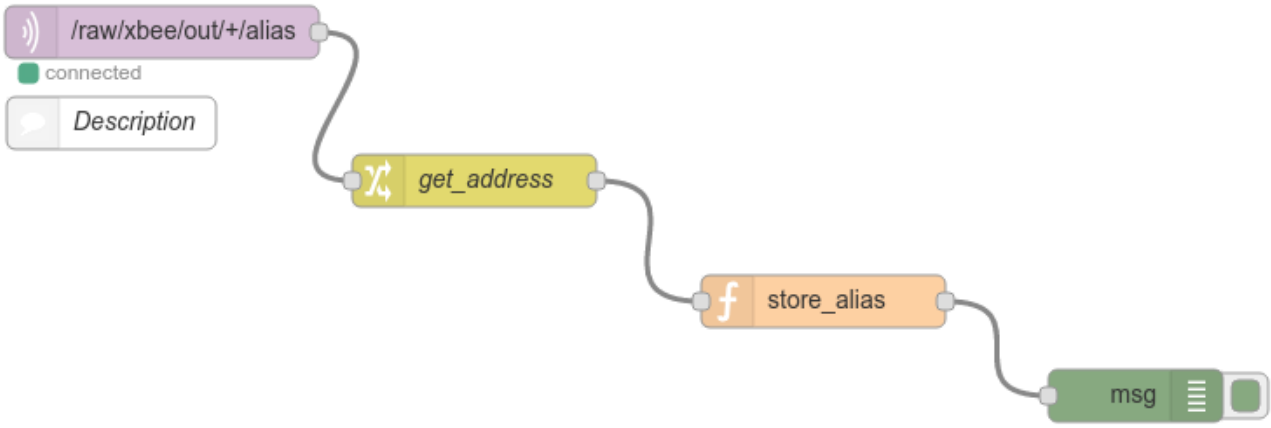
Conecta las actualizaciones emitidas por las entradas analógicas de los XBee a los correspondientes sensores analógicos en Domoticz.

8.5.4 Detect Devices



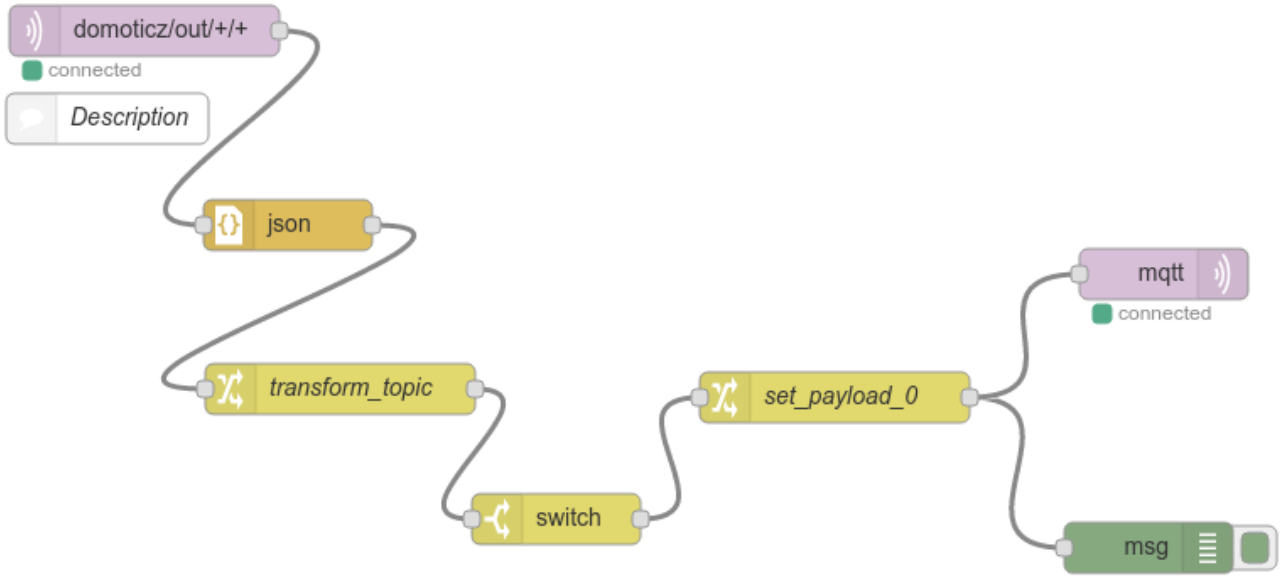
Detecta determinadas configuraciones activas para cada pin de los dispositivos XBee y como consecuencia, crea los sensores/actuadores correspondientes en Domoticz.

8.5.5 Alias



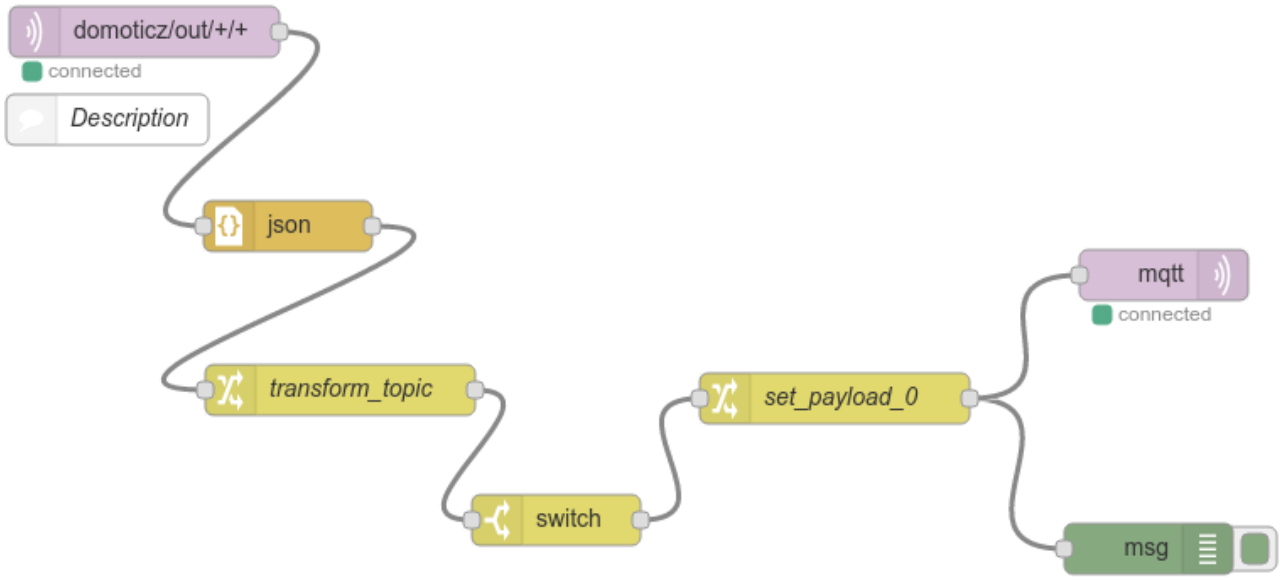
Detecta los alias asociados a las direcciones de los Xbees y los almacena como variables globales para su uso por otros flujos.

8.5.6 Delete



Detecta eventos correspondientes a la eliminación de dispositivos en el control central y desactiva la configuración en XBee estableciendo el pin correspondiente a cero. the configuration of XBee's pins to zero.

8.5.7 Add MQTT Hardware



Detecta la conexión de una pasarela xbee2mqtt y se asegura de que existe un hardware MQTT en Domoticz. Si no existe, lo detecta con una llamada GET a la API JSON.

Bibligrafía

Repositorio del proyecto:

https://github.com/ULL-InformaticaIndustrial-Empotrados/TFG_ZigBee_Embedded_Controller

Pandaboard ES

Board References

http://pandaboard.org/content/resources/references

http://www.omappedia.org/wiki/PandaBoard

LinuxOnArm (PandaBoard)

https://eewiki.net/display/linuxonarm/PandaBoard

https://eewiki.net/display/linuxonarm/PandaBoard_Comments

Tizen IVI Getting Started Guide For PandaBoard

https://wiki.tizen.org/wiki/Tizen_IVI_Getting_Started_Guide_For_PandaBoard

Ubuntu Core Installation

OMAP Ubuntu Core

http://omappedia.org/wiki/OMAP_Ubuntu_Core

SD Configuration

http://omappedia.org/wiki/SD_Configuration

Panda Board Bring Up OMAP Ubuntu Core

http://www.allmost.org/2012/12/panda-board-bring-up-omap-ubuntu-core.html

Ubuntu Core Snappy ported to OMAP4-PANDA board

https://lists.ubuntu.com/archives/snappy-devel/2015-April/000467.html

Running Ubuntu Core on Pandaboard ES

https://groups.google.com/forum/#!topic/pandaboard/W6kzsZwCvj0

Ubuntu Core Installation Tips

https://developer.ubuntu.com/en/snappy/start/installation-tips/

Ubuntu Snappy Core for Gumstix (Prebuilt Images)

https://github.com/gumstix/snappy

Ubuntu Core (All-snaps)

Publication Notes

https://wiki.ubuntu.com/SecurityTeam/PublicationNotes

'New all-snap images available' (list snappy-devel)

https://lists.ubuntu.com/archives/snappy-devel/2016-January/001400.html

All-snaps images (Michael Vogt)

https://people.canonical.com/~mvo/all-snaps/

Fix to 'failed to install: exit status 2'

https://lists.ubuntu.com/archives/snappy-devel/2016-April/001735.html

Snappy Development Tools (zyga)

https://github.com/zyga/devtools/

Snappy Systems

http://bazaar.launchpad.net/~snappy-dev/snappy-hub/snappy-systems/files

ubuntu-device-flash errors for building armhf image

https://lists.ubuntu.com/archives/snapcraft/2016-June/000091.html

Snapd/Snapcraft

Ubuntu Doc

https://developer.ubuntu.com/en/snappy/guides/

Cross Build for amrhf

https://developer.ubuntu.com/en/snappy/guides/cross-build/

Does anyone know how to build snap for ARM architecture? (16.04)

https://lists.ubuntu.com/archives/snapcraft/2016-August/000821.html

Snapcraft

http://snapcraft.io/

Learn to make a Snap

http://snapcraft.io/create/

Snappy, Snapcraft and Interfaces

http://www.zygoon.pl/2016/04/snappy-snapcraft-and-interfaces.html

Única referencia al uso de snapcraft para crear gadgets snaps actuales

https://bugs.launchpad.net/snapcraft/+bug/1560481

Snappy Packages

https://github.com/snappy-packages

Snappy Hacker Useful Link Collection

https://wiki.ubuntu.com/SnappyHackerUsefulLinkCollection

Mir Snaps

https://developer.ubuntu.com/en/snappy/guides/mir-snaps/

Support snap configuration

https://bugs.launchpad.net/ubuntu/+source/snapd/+bug/1596629

How to INSTALL & RUN QML QtWebEngine & QtWebKit on SBC using Yocto / Unable to fetch URL from any source

http://stackoverflow.com/questions/30422646/how-to-install-run-qml-qtwebengine-qtwebkit-on-sbc-using-yocto-unable-to-f

Linux Kernel for Panda

Snapcrafting a kernel

http://blog.sergiusens.org/posts/Snapcrafting-a-kernel/

Patchwork ARM: OMAP4: Basic default configuration for Pandaboard
https://patchwork.kernel.org/patch/1654951/

KernelGitGuide (ubuntu)
https://wiki.ubuntu.com/Kernel/Dev/KernelGitGuide

OMAP FB and DSS on the PandaBoard ES
http://www.crashcourse.ca/wiki/index.php/OMAP_FB_and_DSS_on_the_PandaBoard_ES

[PATCH] ARM: OMAP4: Basic default configuration for Pandaboard
https://groups.google.com/forum/#!topic/pandaboard/2bA5gihN5ws

U-Boot

Das U-Boot
https://wiki.openwrt.org/doc/techref/bootloader/uboot

Das U-Boot Environment
https://wiki.openwrt.org/doc/techref/bootloader/uboot.config

Das U-Boot Commands
http://www.denx.de/wiki/view/DULG/UBoot

Eewiki U-Boot patches
https://github.com/eewiki/u-boot-patches

*** Warning - bad CRC, using default environment
http://www.at91.com/discussions/viewtopic.php/t,25278.html

bad CRC
https://bugs.launchpad.net/ubuntu/+source/goget-ubuntu-touch/+bug/1579767

U-Boot Splash Screen Support
http://www.denx.de/wiki/DULG/UBootSplashScreen

Linux Splash Screen Support
http://www.denx.de/wiki/DULG/LinuxSplashScreen

U-Boot Bitmap support

http://www.denx.de/wiki/DULG/UBootBitmapSupport

Displaying a Splash Screen with U-Boot

https://boundarydevices.com/displaying-a-splash-screen-with-u-boot/

OpenHAB

Sitio Web

http://www.openhab.org/

openHAB on Ubuntu Snappy

https://github.com/openhab/openhab/wiki/Ubuntu-Snappy

OpenHAB bindings

Diaulael XBee Bindings for OpenHAB
https://code.google.com/archive/p/diaoulael-xbee

OpenHAB2 Addons (cdjackson-zigbee)

https://github.com/cdjackson/openhab2-addons/tree/zigbee/addons/binding/org.openhab.binding.zigbee

DIY on XBee Binding (diyxbee)

https://github.com/juri8/openhab/wiki/DIY-on-XBee-Binding

https://community.openhab.org/t/new-binding-for-home-made-hardware/2264

Interfacing XBee with OpenHAB (diaoulael-xbee)

http://technomindshare.blogspot.com.es/2014/04/interfacing-xbee-with-openhab.html

Domoticz

Sitio Web

http://www.domoticz.com/

Domoticz Touchscreen

https://www.domoticz.com/forum/viewtopic.php?t=8678

Domoticz MQTT

https://www.domoticz.com/wiki/MQTT

MQTT Support

https://www.domoticz.com/forum/viewtopic.php?t=838

Domoticz API/JSON URL's

https://www.domoticz.com/wiki/Domoticz_API/JSON_URL%27s

MQTT

Why MQTT

https://learn.adafruit.com/mqtt-adafruit-io-and-you/why-mqtt

XBee to MQTT Gateway

http://tinkerman.cat/xbee-to-mqtt-gateway/

https://code.google.com/archive/p/xoseperez-python-xbee/

ZigBee, XBee and MQTT

http://rijware.com/zigbee-and-mqtt/

MQTT Essentials Part 5: MQTT Topics & Best Practices

http://www.hivemq.com/blog/mqtt-essentials-part-5-mqtt-topics-best-practices

X-Bee (XB24-BUIT-004)

XBee-PRO ZB RF Modules User Guide

http://www.digi.com/resources/documentation/digidocs/PDFs/90000976.pdf

XCTU Software

http://www.digi.com/support/productdetail?pid=3352&type=utilities

XBee v1.1 Adapter files

https://learn.adafruit.com/xbee-radios/download

Video Tutorial XBee: Modo AT, API y Nodos

http://blog.bricogeek.com/noticias/electronica/video-tutorial-xbee-

modo-at-api-y-nodos/

Usando XBee con BeagleBone

http://www.internetdelascosas.cl/2012/09/04/usando-xbee-con-beaglebone/

Python-XBee

Python XBee Documentation

https://python-xbee.readthedocs.io/en/latest/

Python XBee Documentation (GitHub)

https://github.com/nioinnovation/python-xbee/blob/master/docs/source/index.rst

remote_at usage

https://groups.google.com/forum/#!topic/python-xbee/hC9EiZ9L1qU

Node-RED

Node-RED: Documentation

http://nodered.org/docs/

Node-RED: Installation

http://nodered.org/docs/getting-started/installation

Node-RED: Writing Functions

http://nodered.org/docs/writing-functions.html

Domoticz: Node-RED Flows examples

https://www.domoticz.com/wiki/Flows

Proyectos similares

Home Automation System using ZigBee and PandaBoard as a Gateway (HAS-ZP)

http://searchdl.org/public/journals/2014/IJRTET/10/2/4.pdf

Sistema de control remoto para aplicaciones domóticas a través de internet

http://arantxa.ii.uam.es/~jms/pfcsteleco/lecturas/20141107MarioRodriguezCerezo.pdf

Control vía ZigBee de dispositivos de iluminación con protocolo DMX para salas de estimulación multisensorial

https://zaguan.unizar.es/record/5337#

Caretaker Smart Home Server Project

http://www.grappendorf.net/projects/caretaker

Infraestructura de eventos para la Internet de las cosas.

https://ruidera.uclm.es/xmlui/bitstream/handle/10578/4006/TFG_RobertoRequena.pdf

Level Converters

Wifly Adapter Board schema

https://github.com/grappendorf/caretaker-devices/tree/master/wifly-adapter-board

Level-Converter-v10 (SparkFun Electronics)

https://www.sparkfun.com/datasheets/BreakoutBoards/Level-Converter-v10.pdf

Misc

Selecting Controller (MySensors)

http://www.mysensors.org/controller/

Open Source Home Automation Interfaces (comparative)

https://forum.mysensors.org/topic/175/open-source-home-automation-raspberry

Discover the Unknown: Analyzing an IoT Device

https://www.insinuator.net/2016/04/discover-the-unknown-analyzing-an-iot-device/

Diagnose a dead Pandaboard

http://pandaboard.narkive.com/mBd2hEjC/pandaboard-is-dead

PCB Prototype the Easy Way

http://www.pcbway.com/

Running ARM programs under Linux

https://gist.github.com/Liryna/10710751

Using ccache to speed-up the kernel

http://linuxdeveloper.blogspot.com.es/2012/05/using-ccache-to-speed-up-kernel.html

Hachi - Python Library that interacts with XBee

https://github.com/Diaoul/hachi

End-Up solution with MQTT

https://github.com/openhab/openhab/issues/388?source=cc#issuecomment-40284253