Universitat Oberta de Catalunya

uoc.edu

Diseño de una WSN para la estimación del seeing de la cúpula D080, en el Observatorio Astrofísico de Javalambre

Sergio Rueda Teruel
Máster Universitario en Ingeniería de Telecomunicación UOC-URL
Sistemas de Comunicación

Consultor
Raúl Parada Medina

Profesor responsable de la asignatura
Carlos Monzo Sánchez

10 de junio de 2018

A Diego y Alejandro, por las horas de juegos robadas.
A Cristina, sin tu apoyo no lo habría conseguido.

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Diseño de una WSN para la estimación del seeing de la cúpula D080, en el Observatorio Astrofísico de Javalambre*
Nombre del autor:	*Sergio Rueda Teruel*
Nombre del consultor/a:	*Raúl Parada Medina*
Nombre del PRA:	*Carlos Monzo Sánchez*
Fecha de entrega (mm/aaaa):	06/2018
Titulación:	*Máster Universitario en Ingeniería de Telecomunicación UOC-URL*
Área del Trabajo Final:	*Sistemas de comunicación*
Idioma del trabajo:	*Español*
Palabras clave	*WSN* *Seeing* *Temperatura*

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados i conclusiones del trabajo.*

En los observatorios astrofísicos profesionales resulta imprescindible obtener la máxima calidad en las imágenes capturadas, uno de los factores que más afectan a esta calidad, es lo que se conoce como seeing (visibilidad astronómica), que no es más que la distorsión que la turbulencia del aire introduce en las imágenes de los objetos astronómicos que son captadas por los diferentes instrumentos.

Existen diversos factores que contribuyen a la degradación de este seeing, algunos de ellos tienen un origen "natural" como son las condiciones atmosféricas, la altitud de las instalaciones, etc. Mientras que otras, que tienen un origen local, son generadas por los propios sistemas de observación, telescopios, edificios, cúpulas, etc.

Diversos estudios han demostrado que una de las mayores contribuciones al empeoramiento de las condiciones naturales del seeing es debida a las turbulencias que se producen por las diferencias de temperatura entre la superficie de las cúpulas y el aire que las rodea, es lo que se conoce como seeing de cúpula.

Se pretende, determinar si es posible, mediante el diseño de una red de sensores inalámbricos, medir esta diferencia de temperaturas y con ello, cuantificar la contribución que la cúpula que alberga al telescopio de 80cm (D080), del Observatorio Astrofísico de Javalambre (OAJ) aporta al seeing.

De esta manera, una vez que esta degradación fuese determinada, se podrían establecer las acciones oportunas para la minimizar dicha contribución

Abstract (in English, 250 words or less):

In professional astrophysical observatories is essential to obtain the highest quality in the captured images. One of the factors that most affect this quality, is known as seeing (astronomical visibility), that is nothing more than the distortion that the turbulence of the air introduces into the images of the astronomical objects that are captured by the different instruments.

There are several factors that contribute to the degradation of this image quality, some of them have a "natural" origin like atmospheric conditions, the altitude of the facilities, etc., while others have a local origin because they are generated by the observation systems themselves, for instance, telescopes, buildings, domes, etc.

Several studies have shown that one of the greatest contributions to degradation of the natural conditions of the seeing is the turbulence that is produced by the differences in temperature between the surfaces of the domes and the air that surrounds them, it is known as seeing of dome.

It is intended to determine if it is possible, through the design of a wireless sensor network, to measure this temperature difference and, therefore, to quantify the contribution of seeing's degradation, due to the dome that houses the 80cm telescope (D080) in the Astrophysical Observatory of Javalambre (OAJ).

In this way, once this worsening was determined, the appropriate actions could be established to minimize the said contribution.

Índice

1. Introducción 1
1.1. Contexto y justificación del Trabajo 1
1.2. Objetivos del Trabajo 8
1.3. Enfoque y método seguido 9
1.4. Planificación del Trabajo 11
1.5. Breve sumario de productos obtenidos 12
1.6. Breve descripción de los otros capítulos de la memoria 12
2. Estado del Arte 14
2.1. Antecedentes 14
2.2. Medida del seeing en otros observatorios 14
2.2.1. Telescopio Corporación Canadá-Francia-Hawái (1991) 15
2.2.2. Observatorio Gemini (1993) 16
2.2.3. Observatorio Astrofísico de Cagliari (1998) 16
2.2.4. Real Observatorio Greenwich (1998) 17
2.2.5. Observatorio del Roque de los Muchachos. Telescopio William Herschel (1999) 18
2.2.6. Observatorio Astronómico Nacional de San Pedro Mártir (2007) 18
2.2.7. Telescopio Discovery Channel (2007) 19
2.2.8. Instituto Astrofísico de Canarias. Instrumento g-SCIDAR (2008) 20
2.2.9. Gran Telescopio Binocular (2010) 21
2.2.10. Instituto Astronómico Estatal Shtérnberg (2011) 21
2.2.11. Domo Fuji en la Meseta Antártica (2013) 22
2.2.12. Observatorio Gemini (2014) 23
2.2.13. Línea temporal de la evolución en el estudio del seeing local 23
2.2.14. Resumen 23
2.3. Uso de las WSNs en los observatorios astrofísicos 24
2.3.1. Breve descripción de la evolución histórica de las WSNs 24
2.3.2. Aplicaciones basadas en WSNs en observatorios astrofísicos ... 25
2.4. Aportaciones del presente proyecto 25
3. Fundamentos de las redes de sensores 26
3.1. Introducción a las redes de sensores 26
3.2. Arquitectura de un nodo sensor 27
3.3. Factores a tener en cuenta en el diseño de una red de sensores 27
4. Muestra de algunas tecnologías en las WSN actuales 28
4.1. Estándar IEEE 802.15.4 28
4.1.1. Zigbee: 29
4.1.2. 6LoWPAN: 29
4.1.3. WirelessHART 30
4.1.4. ISA100.11a 30
4.2. Bluetooth y Bluetooth Low Energy (BLE) 31
4.3. Z-Wave 31
4.4. ANT 31
4.5. LoRa/LoRaWAN 32
4.6. Wavenis 32
4.7. SigFox 33
4.8. Dash7 33
4.9. EnOcean 33
4.10. RFID Activos 34

4.11. Comparativa entre tecnologías 35
4.12. Tecnología elegida. Justificación 36
4.12.1. Razones tecnológicas 36
4.12.2. Razones prácticas 36
5. Descripción de los elementos utilizados 38
5.1. Hardware 38
5.1.1 Sensores inalámbricos (WSN) 38
5.1.1.1. Tipos de módulos XBee (ZigBee) 38
5.1.1.2. Versión elegida 39
5.1.1.3. Entradas analógicas y conversor AD 40
5.1.1.4. Digi Mesh Kit 40
5.1.1.5. Parametrización de los nodos XBee 41
a) ID. PAN ID (Identificador de red) 41
b) JV. Channel Verification (Verificación del canal) 42
c) CE. Coordinator Enable (Fijar como coordinador) 42
d) SH y SL. Serial Number High y Low (Número de serie del nodo) 43
e) DH y DL. Destination Number Hig y Low (Dirección del nodo destino) 43
f) NI. Node Identifier (Identificador del nodo) 44
g) AP. API Enable (Habilita el modo de transmisión basado en paquetes) 44
h) SM. Sleep Mode (Modos de reposo) 44
i) SP. Sleep Period (Periodo de reposo) 45
j) SN. Number of Cyclic Sleep Periods (Número de ciclos de reposo) 45
k) ST. Time Before Sleep (Tiempo antes de dormir) 46
l) SO. Sleep Options (Opciones de sueño) 46
m) SO. Parámetros de diagnóstico. 46
5.1.2 Sensores de temperatura. 47
5.1.2.1 Requerimientos previos 47
5.1.2.2 Sensor TPM36 48
5.1.2.3 Sensor LMT84 48
5.1.2.4 Termistor 49
5.1.2.5 Elección del sensor 49
5.1.2.6 Cálculos y calibración de los sensores 50
5.1.3 Fabricación de la PCB 51
5.1.4 Gateway. Nodo colector 53
5.1.4.1 PC de sobremesa 53
5.1.4.2 Raspberry pi 2 model b 53
5.1.5 Sistema de alimentación. Opciones 54
5.1.5.1 Baterías alcalinas 55
5.1.5.2 Baterías de litio 55
5.1.5.3 Baterías de Níquel-Metal-Hidruro 55
5.1.5.4 Batería elegida 55
5.1.6 Diseño caja estanca para albergar la electrónica 57
5.1.7 Esquema físico 58
5.2 Software 59
5.2.1 Entorno de desarrollo nodos WSN (radio) 59
5.2.1.1 Digi XCTU 59

5.2.2 Gateway nodo colector. Parte comunicación 60
5.2.2.1 Python 3 (Pycharm) 61
5.2.3 Gateway nodo colector. Tratamiento de datos 67
5.2.3.1 InfluxDB 67
5.2.3.2 Telegraf 67
5.2.3.3 Grafana 68
5.2.4 Esquema conceptual del conjunto 69
5.2.5 Diseño de las PCB. EasyEDA 70
5.2.6 Software de diseño 3D de la caja. Tinkercad 70
6. Sistema prototipo 71
6.1 Pruebas de viabilidad en prototipo inicial 71
6.2 Integración en las PCBs 72
6.3 Pruebas de conjunto en laboratorio 73
7 Fase de despliegue 75
8 Valoración económica del trabajo 80
8.1 Kit de iniciación y prototipo 80
8.2 Fabricación de nodos 80
8.3 Horas de Ingeniería 82
8.4 Material de backup 82
8.5 Coste total 83
9 Análisis de los resultados 83
10 Conclusiones 85
11 Bibliografía 89
12 Anexos 97
12.1 Esquema eléctrico 97
12.2 Módulos Python3 desarrollados 98
12.2.1 discover_devices_xbee.py 98
12.2.2 get_fwhm.py 99
12.2.3 gui.py 101
12.2.4 read_local_params_xbee.py 106
12.2.5 read_node_config_file.py 110
12.2.6 read_node_list.py 115
12.2.7 read_remote_ADC_xbee_multihilo.py 115
12.2.8 read_remote_params_xbee.py 118
12.2.9 read_sys_config.py 122
12.2.10 send_data_to_telegraf.py 122
12.2.11 write_local_params_xbee.py 123
12.2.12 write_remote_params_xbee.py 127
12.2.13 Ejemplo archivo configuración nodo remoto. 130
12.2.14 list_of_nodes.ini 131
12.2.15 sys_config.ini 132
12.3 Manual básico de instalación 132
12.3.1 Software de adquisición (Python3) 132
12.3.2 Software de almacenamiento y monitorización 133
12.4 Manual de usuario 135
12.4.1 Software de adquisición y configuración (Python3) 135
12.4.2 Software de monitorización Grafana 137

Lista de figuras

Ilustración 1. Telescopio principal del OAJ. JST/T250. 1
Ilustración 2. Telescopio auxiliar del OAJ. JAST/T80. 2
Ilustración 3. Turbulencia del aire que contribuye al seeing de espejo 4
Ilustración 4. Turbulencia del aire interior que contribuye al seeing local 4
Ilustración 5. Turbulencia del aire exterior que contribuye al seeing local 4
Ilustración 6. Idea conceptual ubicación de los nodos interior de D080 6
Ilustración 7. Idea conceptual ubicación de los nodos en el exterior de D080. .. 7
Ilustración 8. Vista aérea del Observatorio Astrofísico de Javalambre..... 7
Ilustración 9. Planificación del trabajo del TFM ……..11
Ilustración 10. Observatorio CFHT en Hawái (Mauna Kea) 15
Ilustración 11. Observatorio Gemini Norte en Hawái (Mauna Kea)..... 16
Ilustración 12. Observatorio de Cágliari 16
Ilustración 13. Real Observatorio de Greenwich 17
Ilustración 14. Telescopio William Herschel) 18
Ilustración 15. Observatorio San Pedro Mártir..... 18
Ilustración 16. Telescopio Discovery Channel..... 19
Ilustración 17. Observatorio del Roque de los Muchachos (IAC) 20
Ilustración 18. Gran Telescopio Binocular..... 21
Ilustración 19. Observatorio de las montañas del Cáucaso 22
Ilustración 20. Telescopio DIMM en Domo Fuji (Antártida) 22
Ilustración 21. Esquema tipo de una red de sensores…….26
Ilustración 22. Arquitectura fundamental de un nodo WSN..... 27
Ilustración 23. Módulo Digi XBee elegido..... 40
Ilustración 24. Elementos que integran el Digi Zbee Zigbee Mesh Kit 41
Ilustración 25. Elementos que forman una red ZigBee (XBee)..... 43
Ilustración 26. Curva de ajuste a los valores de resistencia en los NTC 51
Ilustración 27. Vista superior de la PCB diseñada..... 52
Ilustración 28 Vista inferior de la PCB diseñada..... 53
Ilustración 29. Excel para el cálculo de vida de la batería..... 56
Ilustración 30. Caja diseñada para alojar la electrónica 58
Ilustración 31. Esquema físico de todos los componentes..... 58
Ilustración 32. Secuenciación temporal de todo el proceso 59
Ilustración 33. Software XCTU 60
Ilustración 34. GUI desarrollado en Python3 (Lectura parámetros)..... 63
Ilustración 35. GUI desarrollado en Python3 (Descubrimiento)..... 63
Ilustración 36. Interface Web Grafana..... 69
Ilustración 37. Esquema lógico de todo el sistema..... 70
Ilustración 38. Diseño de la caja para la electrónica. Tinkercad..... 71
Ilustración 39. Diseño de la tapa para la caja de la electrónica. Tinkercad 71
Ilustración 40. Primer prototipo de un nodo sensor..... 71
Ilustración 41. Prototipo final de un nodo sensor integrado en placa. 72
Ilustración 42 Fotografía de la PCB con todos los componentes 72
Ilustración 43. Grupo de nodos en laboratorio, antes de pintar..... 73
Ilustración 44. Vista de todos los nodos finalizados 75
Ilustración 45. Esquema de colocación de los nodos..... 76
Ilustración 46. Imágenes del desplieque real dentro de la D080..... 77
Ilustración 47. Cuadros eléctricos fijos para el control de la cúpula 78

Ilustración 48. Imagen del sistema de monitorización 79
Ilustración 49. Captura de toda la noche de observación..... 84
Ilustración 50. Captura del periodo de observación en el nodo REMOTO_4 ... 84
Ilustración 51. Captura del periodo de observación en el nodo REMOTO_7 ... 85
Ilustración 52. Captura del periodo de observación en el nodo REMOTO_9 ... 85
Ilustración 53. Inteface de configuración Grafana 134
Ilustración 54. Ejemplo de query Grafana 135
Ilustración 55. Interface gráfica de configuración de los nodos 136

Glosario

ADC – *Analog to Digital Converter*
AES – *Advanced Encryption Standard*
ARPANET - *Advanced Research Projects Agency Network*
BLE – *Bluetooth Low Energy*
CCD – *Charge-Coupled Device*
CFHT – *Canada-France-Hawaii Telescope*
CSIC – Centro Superior de Investigaciones Científicas
CSS – C*hirp Spread Spectrum*
D080 – *Dome T080 Telescope*
D250 – *Dome T250 Telescope*
DCT – *Discovery Channel Telescope*
DIMM – *Differential Image Motion Monitor*
DSN – *Distributed Sensor Network*
DSSS – D*irect Sequence Spread Spectrum*
FFD – *Full Function Devices*
FHS – *Frequency Hopping Spread Spectrum*
FoV – *Field of View*
FWHM – *Full Width Half Maximum*
GTC-GRANTECAN– Gran Telescopio de Canarias
IAC – Instituto Astrofísico de Canarias
ICTS – Infraestructura Científica y Tecnológica Singular
IEEE – *Institute for Electrical and Electronics Engineers*
IETF – *Internet Engineering Task Force*
INAF-OCA - Istituto Nazionale di Astrofisica - Observatoire de la Côte d'Azur
IoT – *Internet of Things*
ISA – *International Society of Automation*
JAST – Javalambre *Auxiliary Survey Telescope*
JOSE – *Joint Observatories Seeing Evaluation project*
JPCam– *Javalambre Physics Camera*
JST – Javalambre *Survey Telescope*
LTS – *Long Time Support*
MAC – *Medium Access Control*
MMT – *Micro Mount*
OAJ – Observatorio Astrofísico de Javalambre
P2P – *Point to Point*
PC – *Personal Computer*
PCB – *Printed Circuit Board*
PHY – *Physical*
PLA – Poli-ácido Láctico
PSF – *Point Spread Function*
PSS – *Primary Synchronization Signal*
PWM – Pulse-width modulation
RF – Radio Frecuencia
RFD – *Reduced Function Devices*
RFID – *Radio Frequency Identification*
SMT – *Surface Mount*

SOSUS– *Sound Surveillance System*
T080 – Telescopio 80cm
T250 – Telescopio 250cm
TFM – Trabajo Fin de Máster
TH – *Through Hole*
UNAM – Universidad Nacional Autónoma de México
UTC – *Universal Time Clock*
UWB – *Ultra Wide Band*
VPN – *Virtual Private Network*
WSN – *Wireless Sensor Network*
XCTU – *XBee Configuration and Test Utility software*

1.Introducción

A lo largo de los siguientes apartados, se tratará de dar una visión global del contexto y el alcance de este trabajo, así mismo, se indicarán los objetivos que se persiguen y la planificación para lograrlos. Finalmente se detallarán los productos obtenidos y un breve resumen de los capítulos posteriores.

1.1. Contexto y justificación del Trabajo

El Observatorio Astrofísico de Javalambre (OAJ) es una Infraestructura Científica y Técnica Singular (ICTS) astronómica española ubicada en la Sierra de Javalambre, en Teruel, concretamente en el "Pico del Buitre" (40º02'30.58" Norte, 01º00'58.58" Oeste), a 1957 metros de altitud, en la localidad de Arcos de las Salinas, el objetivo fundamental del observatorio es la realización de grandes cartografiados astronómicos. El OAJ consta principalmente de dos telescopios profesionales de gran campo de visión, *Field of view* (FoV), con calidad de imagen en todo el campo: el *Javalambre Survey Telescope* (JST/T250), un telescopio de 2.55 m de apertura, con un campo de visión 3 grados cuadrados, y el *Javalambre Auxiliary Survey Telescope* (JAST/T80), un telescopio de 80 cm de apertura, con un campo de visión de 2 grados cuadrados. Ambos telescopios están equipados con cámaras panorámicas de última generación cuyos sensores o CCDs de gran formato (en inglés, *Charge-coupled device* [1]) junto a los filtros ópticos especialmente diseñados, forman un conjunto único para realizar una cartografiado del Universo en todo el rango del espectro óptico sin precedentes.
La cámara que equipa el JST/250 es conocida como JPCAM y fue, en el momento de su fabricación, la cámara más grande del mundo (hoy ocupa el segundo lugar) con 14 detectores CCD de gran formato de 9200x9200 píxeles de 10µm, por otra parte, el sistema de filtros está formado por 54 filtros estrechos (~12.5nm) espaciados entre 350nm y 1000nm y 2 filtros de banda ancha para obtener precisiones de *redshift* fotométricos (corrimiento al rojo [2]) que nadie ha conseguido hasta ahora.

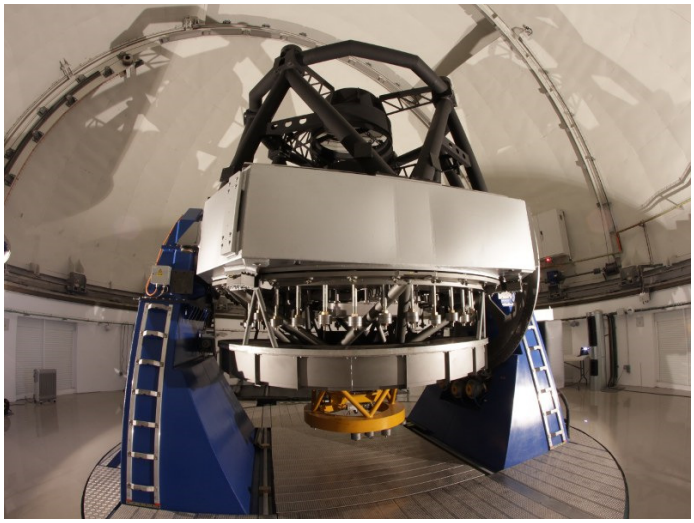


Ilustración 1. Telescopio principal del OAJ. JST/T250.

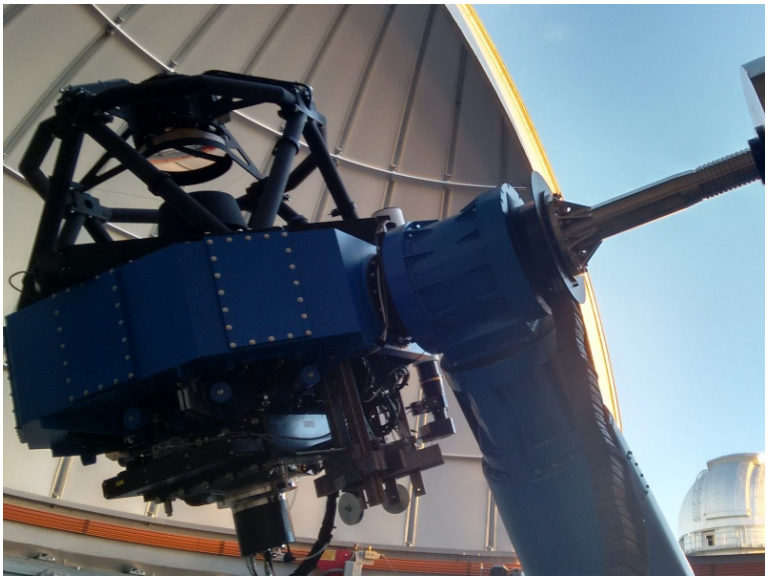


Ilustración 2. Telescopio auxiliar del OAJ. JAST/T80.

Tal y como se ha comentado, uno de los requerimientos más importantes para el observatorio, es obtener la máxima calidad en las imágenes, entendida como la ausencia de distorsión en la imagen telescópica en el intervalo de observación. Esta calidad de imagen o calidad óptica viene afectada por diversos factores, algunos de ellos provienen de las propias características de los telescopios e instrumentos utilizados, como, por ejemplo, la calidad del guiado del telescopio, calidad de los espejos, de las lentes, sistemas de enfocado, variaciones en la sensibilidad de los píxeles individuales de los CCDs, etc.
Otras se deben a condiciones del cielo, como contaminación lumínica, extinción (absorción y dispersión de la radiación electromagnética emitida por los objetos debido a gas, polvo, etc.) Nubes, presencia de la Luna, etc.
Y otras se deben a efectos relacionados con la turbulencia del aire que la luz atraviesa, así, es bien conocido que el índice de refracción del aire cambia con su densidad, y que dos cuerpos de aire a diferentes temperaturas y/o humedades crean un límite de refracción que dobla la luz [3].
A efectos prácticos y con el afán de simplificar, podemos decir que las variaciones temperatura y humedad del aire, crean turbulencia en el mismo, lo que hace que los objetos astronómicos centellen y formen imágenes borrosas, poniendo un límite a la capacidad de los telescopios e instrumentos para resolver objetos astronómicos.

De esta manera, una imagen inmóvil y ópticamente perfecta, indica un seeing excelente, mientras que una imagen que cambia rápidamente genera una distorsión e indica un seeing pobre, más adelante se darán algunos valores, aunque, para hacernos una idea de la importancia de este fenómeno, cabe recordar que la NASA invirtió 2800 millones de dólares estadounidenses en el Telescopio Espacial Hubble para obtener un *seeing* 0", puesto que la luz que recoge no atraviesa la atmosfera.

El *seeing* se suele expresar en segundos de arco (aunque también se utilizan otras escalas como la de Pickering o de Antoniadi [4]) y sus valores, en lugares

no aptos para la observación, fácilmente varían entre 2", 5", 10" sin embargo, en los emplazamientos de los observatorios profesionales este valor mejora considerablemente, y se llega a ese *seeing* excelente antes mencionado, así, por ejemplo, el OAJ, presenta un *seeing* mediano de 0.70". Este valor es igual al de otros observatorios como el Observatorio del Roque de los Muchachos en La Palma, el Observatorio del Teide en Tenerife [5] y mejor que el del otro gran observatorio en la península, Observatorio de Calar Alto en Almería, con un seeing medio de 0.90" [6].
Fuera de este país, tomamos por ejemplo dos de los observatorios más importantes del mundo con valores muy similares, así por ejemplo el Observatorio de Paranal en Chile presenta un seeing mediano de 0.66" [7] y el Observatorio de Mauna Kea en Hawaii 0.43" [8] .*(Indicar que estos valores varían ligeramente según el año bajo estudio).*

Una de las formas comunes de medir el *seeing* de un emplazamiento, suele ser mediante el uso de un monitor diferencial de imagen, DIMM [9] (por sus siglas en inglés, *Differential Image Motion Monitor*), en el cual la luz se hace pasar por dos aperturas idénticas separadas por una distancia *d*, con el objetivo de producir dos imágenes gemelas de una estrella con el mismo telescopio. El cono de luz que produce cada una de estas aperturas es separado mediante la utilización de un prisma en una de ellas formando por tanto dos imágenes separadas en el plano focal del telescopio. El movimiento relativo de ambas imágenes en el plano focal representa inclinaciones locales del frente de onda. Existen factores que afectarán por igual a las imágenes producidas por ambas aperturas, como, por ejemplo, las vibraciones del telescopio, sin embargo, hay otros factores que afectan de forma diferente a cada una de las aperturas, como la turbulencia atmosférica.
Así la varianza del movimiento longitudinal (σ_l^2) y transversal (σ_t^2) de una imagen diferencial viene dada por [9]:

$$\sigma_l^2 = 2\lambda^2 r_0^{-5/3}[0.179D^{-1/3} - 0.0968d^{-1/3}]$$
$$\sigma_l^2 = 2\lambda^2 r_0^{-5/3}[0.179D^{-1/3} - 0.145d^{-1/3}]$$

Válido para d> 2D, donde D es la apertura del telescopio, r_0 es el parámetro de Fried y λ es la longitud de onda característica.
Así, midiendo las varianzas, es posible calcular r_0 y de allí, obtener el seeing característico (ε_{FWHM}):

$$\varepsilon_{FWHM} = 0.98\ \frac{\lambda}{r_0}\ [10]$$

Dentro de este *seeing*, está lo que se conoce como *seeing* local, que es la contribución que la propia instalación y elementos de observación realizan al *seeing* global. Una revisión a la literatura reciente identifica tres fuentes principales del *seeing* local [11].

- ***Seeing* de espejo**, que queda fuera del objeto del presente trabajo, y se debe a las diferencias térmicas entre la superficie del espejo primario del telescopio y el aire que lo rodea.

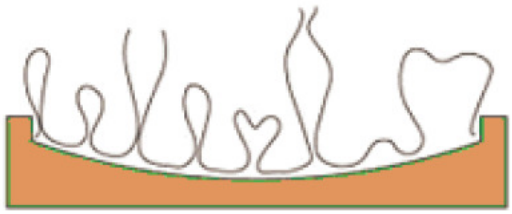


Ilustración 3. Turbulencia del aire que contribuye al seeing de espejo [11]

- ***Seeing* de cúpula interior**, surge de las diferencias en temperatura entre la cara interior de la cúpula, los distintos elementos y la temperatura del aire dentro de la misma.
  $\varepsilon_{CupulaInt} = \mathbf{0.10} \cdot \boldsymbol{\Delta T_{c-a}}$ Donde $T_{c\text{-}a}$ es la diferencia de temperatura entre la superficie interior de la cúpula y la temperatura del aire ambiente interior.

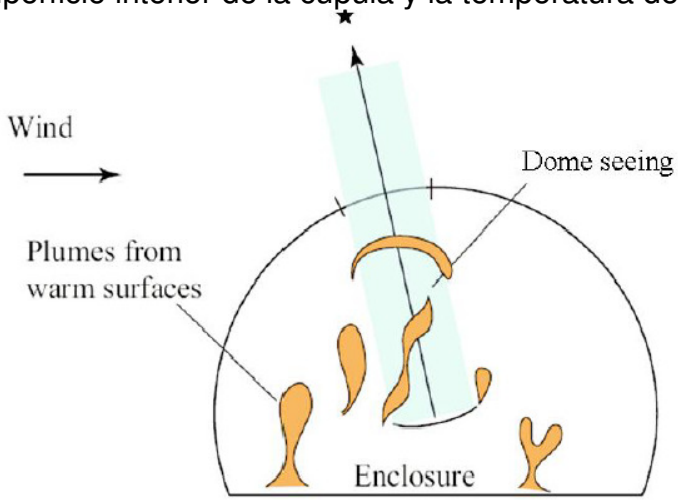


Ilustración 4. Turbulencia del aire interior que contribuye al seeing local [11]

- ***Seeing* de cúpula exterior**, la turbulencia aquí se origina debido a las diferencias de temperatura entre la superficie exterior de la cúpula y la temperatura local del aire que la rodea.
  $\varepsilon_{CupulaExt} = \mathbf{0.12} \cdot \boldsymbol{\Delta T_{a-s}}$ Donde $T_{a\text{-}s}$ es la diferencia de temperatura entre la superficie exterior de la cúpula y la temperatura del aire ambiente que la rodea.

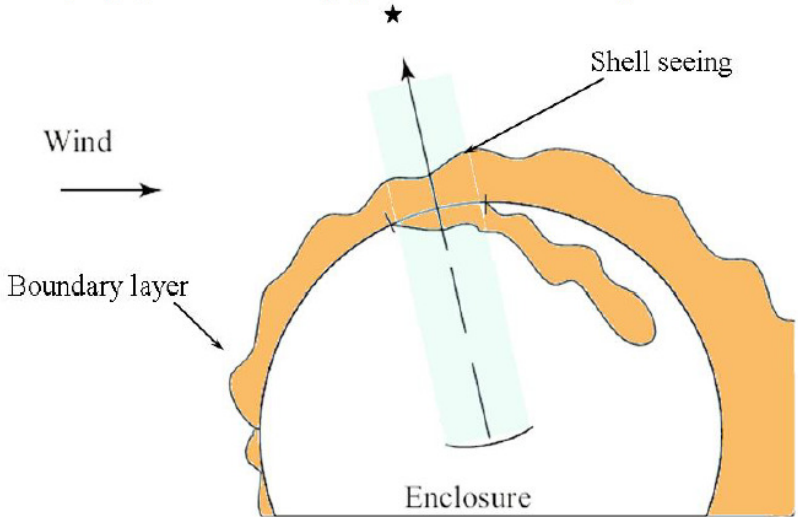


Ilustración 5. Turbulencia del aire exterior que contribuye al seeing local [11]

Evidentemente, en los observatorios profesionales, el estudio de este seeing local adquiere una gran importancia, puesto que una vez que sea determinado, se podrán tomar las medidas correctivas para su minimización en aras de la consecución de la mayor calidad de imagen posible.

En el OAJ, actualmente existen un conjunto de monitores para medir la calidad del cielo, entre ellos un monitor DIMM de *seeing*, que permite cuantificar la contribución total del *seeing* debida a la turbulencia, pero que, sin embargo, no permite discretizar la contribución local que las propias cúpulas introducen en la degradación de este *seeing*, esto es así debido, entre otras cosas, a que el monitor de *seeing* no está ubicado en la misma instalación que los telescopios, por ello este valor se toma como referencia o valor objetivo, sin embargo la imagen al estar afectada por el seeing local obtiene peor calidad óptica, esta degradación en la calidad óptica se ve reflejada en el empeoramiento de la función de dispersión del punto (PSF del inglés, *Point Spread Function* [12]) y a la anchura a media altura (FWHM del inglés, *Full Width Half Maximun* [13]) de cuyos valores se puede deducir la cantidad de degradación total que afecta a la imagen. (El procedimiento para la obtención de este empeoramiento se basa en el cálculo de la PSF a través de la FWHM, en los centros astrofísicos se utiliza un software llamado SEXtractor [14], que mediante cálculos de distribución de la energía en las imágenes, es decir, el número de fotones por pixel, es capaz de obtener dicha FWHM, una vez obtenido este valor de FWHM y conociendo el valor de escala de placa de los CCDs [15] se puede determinar la PSF, esta PSF es por tanto la degradación total de la imagen y que idealmente debería coincidir con el seeing del emplazamiento, evidentemente esto no es así debido a la distorsión que como hemos dicho introducen todos los elementos implicados en la observación)

Siguiendo el razonamiento anterior, midiendo las variaciones de temperatura entre diferentes puntos, esto es, entre las superficies de la cúpula y el ambiente que la rodea podemos ser capaces, viendo la FWHM de las imágenes, de establecer una relación entre esta variación de la temperatura y la variación en la calidad de imagen, y, por tanto, determinar la contribución de la cúpula al seeing local

Hasta la fecha, a la hora de acometer esta tarea, una de las dificultades más importantes que se presentan es que las cúpulas en realidad son mecanismos móviles ya que poseen un movimiento azimutal no limitado, y esclavo del desplazamiento del telescopio, es decir, la rendija o ventana de la cúpula debe rotar, para permitir, en todo momento la perfecta visión del telescopio sin introducir obstáculos en el camino óptico.
Este movimiento, hace que la colocación de diferentes sistemas de medición, en diferentes puntos de la superficie de las cúpulas no sea trivial, debido a la necesidad de alimentación, conectividad y robustez frente a los movimientos.
Otra de las dificultades que se presentan es que los elementos de medición utilizados no deben generar calor, puesto que ellos mismos serían fuentes de turbulencia y, por tanto, se debe limitar el consumo a lo mínimo necesario.
Además, indicar que los sistemas de medición no deben emitir luz, ni ningún tipo de radiación electromagnética que pueda ser captada por los instrumentos.

Mediante el presente proyecto, se pretende superar las dificultades anteriormente planteadas. Para ello, se presenta una solución para la determinación del seeing de la cúpula D080, que es la cúpula que alberga al telescopio JAST/T80, que perfectamente podría ser extrapolada al resto de cúpulas, no solo de este, sino de otros observatorios. Se plantea la instalación de una red de sensores inalámbricos, WSN (del inglés, *Wireless Sensor Network*), distribuidos, de forma que permitan realizar mediciones periódicas de los valores de temperatura en diferentes puntos, de esta forma, mediante las diferencias de temperatura se podrá estimar el *seeing* local de cúpulas, que será contrastado con el *seeing* determinado por la FWHM de las imágenes capturadas. Indicar, además, que se ha de considerar, tal y como se ha comentado, que todos los sensores utilizados, los microcontroladores y los protocolos de comunicación deberán cumplir con el requerimiento de bajo consumo, para por una parte evitar ser puntos de generación de turbulencia por calentamiento de los componentes y por otra, para maximizar su autonomía y eficacia.

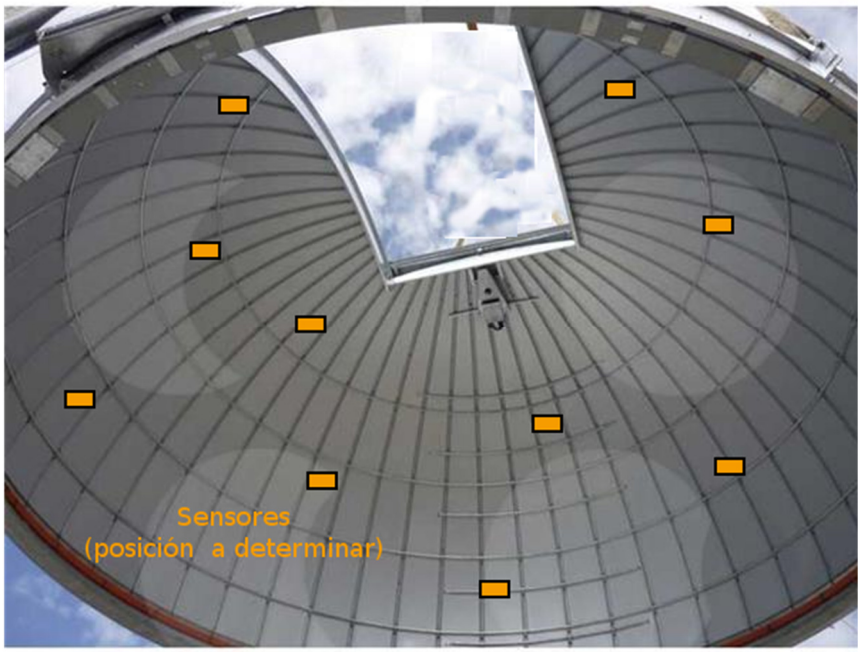


Ilustración 6. Idea conceptual que muestra la ubicación de los nodos interior de la cúpula D080.

Ilustración 7. Idea conceptual que muestra la ubicación de los nodos en el exterior de la cúpula D080.

Se estima que, con 10 nodos, capaces cada uno de tomar temperaturas en, al menos, 2 puntos diferentes, será suficiente para llevar a cabo con éxito el objetivo del proyecto. (Inicialmente, uno de los nodos será colector por lo que su misión no será la de recogida de datos).
Esta estimación se hace realizando una aproximación conservadora, se pretende fijar un prepuesto modesto (400€-500€ aprox.), pero que permita obtener valores útiles, y ver si la relación entre la inversión realizada y lo que aporta al proyecto resulta interesante. En caso de ser así, este trabajo podría ser el pistoletazo de salida para la definición de nuevos proyectos en el OAJ que nos permitieran no sólo mejorar lo obtenido en la D080, sino también ser implementado en la cúpula del telescopio mayor, D250, cuyas dimensiones son considerablemente superiores.
Indicar también, que el hecho de que, como más adelante se mencionará, los sensores sean móviles, permitiría en un futuro y en un plazo de tiempo mayor que lo que abarca este TFM, un estudio zonal, es decir, ir estudiando secuencialmente la contribución al *seeing* de pequeñas zonas de cúpula con una alta densidad de sensores por zona.

Finalmente señalar que la cúpula D080, fue fabricada por la empresa estadounidense Ash-Dome, e instalada en el observatorio a finales del 2011, tiene una estructura semiesférica, con un diámetro de 6.25m (la distancia entre dos nodos nunca superará este valor), un peso aproximado de 2.5 toneladas y una rendija o ventana de 1.8m de ancho, y tal y como se ha dicho, está dotada de un sistema que le permite girar 360º en cualquier dirección, sin límite.

Ilustración 8. Vista aérea del Observatorio Astrofísico de Javalambre.OAJ.

1.2. Objetivos del Trabajo

El fin primordial del presente trabajo consiste en determinar si la aproximación al estudio del *seeing* local en los observatorios astronómicos mediante el uso de redes de sensores es viable, para ello se pretende implementar una WSN que permita estudiar la correlación existente entre las variaciones de temperatura entre las superficies de la D080, el ambiente en torno a ellas, y las variaciones del en la calidad de las imágenes obtenidas. Los objetivos fundamentales por tanto son:

- Análisis de las opciones actuales en las redes de sensores inalámbricos que derive en la elección del modelo más adecuado.

- Fabricación de nodos de sensores inalámbricos que permitan la medición de temperatura en diferentes puntos. Estos nodos serán fácilmente instalables y des-instalables y la red de sensores deberá soportar la inclusión o eliminación de algunos de sus nodos "en caliente" sin la pérdida de su funcionalidad. La fabricación de estos sensores abarcará, entre otras, dos fases bien diferenciadas:

    - Fabricación de nodos prototipo que permitan comprobar la viabilidad del proyecto.

    - Fabricación de nodos finales en una placa de circuito impreso y que, tal y como se ha dicho, permitan ser instalados y desinstalados con gran facilidad en diferentes puntos estratégicos para la obtención de datos útiles.

    - Despliegue en el entorno real, esto es, la cúpula D080 del OAJ, este despliegue se iniciará en la parte interior de la estructura, cuyas condiciones ambientales son menos severas, para que una vez obtenidas las medidas, se proceda a su instalación en la superficie exterior de la cúpula dónde las condiciones climáticas son más duras al tratarse de un lugar de alta montaña.

    - Viabilidad del resultado, como método para obtener la contribución del *seeing* de cúpula y que pueda ser extrapolable a otras cúpulas en distintos lugares y de diferentes dimensiones.

- Diseño del sistema de recogida de datos, este debe permitir que dichos datos sean relacionados con los proporcionados por el observatorio (calidad de las imágenes). Se plantea la existencia de un nodo colector que conectado o bien a un PC o a una *raspberry pi,* sea capaz de ingresar las mediciones en una base de datos para su posterior monitorización y análisis.

- Análisis económico de la solución, tal y como se ha dicho se parte de un presupuesto limitado, unos pocos cientos de euros. (400€-500€ aprox).

Finalmente destacar que el proyecto, por tanto, tiene dos vertientes relacionadas, por un lado, una vinculada directamente con la investigación sobre las redes de sensores y su aplicación práctica en los observatorios astrofísicos y por otro, otra claramente industrial relacionada con la elaboración de un producto para la realización una tarea específica.

1.3. Enfoque y método seguido

Existen otras aproximaciones a la utilización de sensores como serían el software de modelización térmica [16], o incluso la utilización de monitores DIMM en el interior de la cúpula, esto es posible dado que la montura del telescopio T080, está diseñada para soportar la instalación de un telescopio menor (*piggyback telescope* [17]) para tareas de guiado, gracias a esa capacidad sería posible la instalación de un monitor DIMM para obtener directamente el *seeing* de cúpula. No obstante, estas soluciones serían mucho más caras y/o mucho más complejas de implementar, el software de modelización térmica tiene un coste asociado y, por otra parte, llevar alimentación y conectividad al telescopio *piggyback* no es en absoluto trivial, por no mencionar todas las tareas de puesta a punto que requiere un telescopio y la instrumentación profesional.

Por tanto, tal y como se ha dicho, la idea general del proyecto es la creación de una red de sensores inalámbrica que permita medir la diferencia de temperatura en los puntos elegidos y derivar de ellas el deterioro del *seeing* atmosférico.
Se pretende, por consiguiente, la realización de un producto nuevo, que como es una red de nodos que se conectarán entre sí vía inalámbrica, que serán fácilmente instalables y desinstalábles, que periódicamente tomarán datos de temperatura y que serán enviados a un nodo recolector central.
Este nodo recolector aportará también conectividad con el observatorio, es decir, su función no es sólo la de colector sino que también realiza tareas de *gateway* entre el sistema de sensores y el observatorio, para ello se plantea la utilización del propio equipo de control conectado al colector, esto es, un PC o una *raspberry pi* y de una base de datos como el sistema óptimo para la recolección de las medidas, puesto que permitirá su análisis posterior y proporciona gran interoperabilidad con multitud de plataformas diferentes.

Por otra parte, el proyecto se abordará en fases, como se recoge más adelante en la planificación del trabajo, pero en una visión general podríamos concluir en los siguientes acercamientos para la resolución del problema planteado.
En primer lugar, se realizará una simulación de la solución en laboratorio con la creación de una red de sensores, tratando de reproducir todos los aspectos posibles que luego se darán en el entorno real, distancias, movimiento, vibraciones, etc. En este punto requiere especial importancia el análisis de dos puntos clave:

- Análisis del sistema de radio elegido, protocolo, controlador, funcionamiento, configuración, limitaciones, etc.

- Análisis del sistema de sensado, los sensores elegidos deberán poder trabajar con los rangos de temperatura que se dan en la sierra de

Javalambre, y con la resolución y precisión necesarias para la tarea que se pretende acometer.

Una vez que la simulación de la solución haya quedado validada se trabajará en el diseño del sistema colector de datos, que debe realizar las tareas anteriormente especificadas.

Finalmente, una vez finalizado de forma exitosa las fases anteriores se plantea el diseño y creación de los nodos definitivos en placas de circuito impreso que faciliten su posterior instalación. Una vez que los nodos sean fabricados se procederá a su despliegue en la cúpula D080 del OAJ.

1.4. Planificación del Trabajo

Id	Nombre de tarea	Duración	Comienzo	Fin
1	**Comienzo PEC1 (Definición TFM)**	**12 días**	**jue 22/02/18**	**mié 07/03/18**
2	Definir contexto, objetivo, enfoque, etc	6 días	jue 22/02/18	mié 28/02/18
3	Puntos PEC1 (planficación, estructura capítulos, etc)	6 días	jue 01/03/18	mié 07/03/18
4	**Comienzo PEC2 (Estudio trabajos relacionados)**	**10 días**	**jue 08/03/18**	**lun 19/03/18**
5	**Estudio estado del arte**	**4 días**	**jue 08/03/18**	**lun 12/03/18**
6	Estudio estado del arte WSN	3 días	jue 08/03/18	sáb 10/03/18
7	Análisis de los sensores temperatura	1 día	lun 12/03/18	lun 12/03/18
8	**Analisis del sistema colector datos**	**3 días**	**mar 13/03/18**	**jue 15/03/18**
9	Definición del hardware	2 días	mar 13/03/18	mié 14/03/18
10	Definición de software	1 día	jue 15/03/18	jue 15/03/18
11	**Análisis viavilidad técnica**	**3 días**	**vie 16/03/18**	**lun 19/03/18**
12	Estudio del sistema de alimentación	2 días	vie 16/03/18	sáb 17/03/18
13	Análisis y soluciones a puntos críticos	2 días	vie 16/03/18	sáb 17/03/18
14	Analisis integración completa	1 día	lun 19/03/18	lun 19/03/18
15	**Análisis viabilidad económica**	**1 día**	**lun 19/03/18**	**lun 19/03/18**
16	**Comienzo PEC3 (Producto del TFM)**	**54 días**	**mar 20/03/18**	**dom 20/05/18**
17	**Adquisición material para prototipos**	**4 días**	**mar 20/03/18**	**vie 23/03/18**
18	Material nodos prototipo	4 días	mar 20/03/18	vie 23/03/18
19	Material desarrollo en pcb prototipos	4 días	mar 20/03/18	vie 23/03/18
20	**Test laboratorio prototipos (3 nodos)**	**4 días**	**sáb 24/03/18**	**mié 28/03/18**
21	Instalación hardware	2 días	sáb 24/03/18	lun 26/03/18
22	Pruebas de comunicación entre nodos	2 días	mar 27/03/18	mié 28/03/18
23	**Posibles mejoras**	**2 días**	**jue 29/03/18**	**vie 30/03/18**
24	**Diseño y fabricación circuito electronico**	**8 días**	**sáb 31/03/18**	**lun 09/04/18**
25	Diseño placa mediante software PCBs	3 días	sáb 31/03/18	mar 03/04/18
26	Fabricación de la placas pcb	6 días	sáb 31/03/18	vie 06/04/18
27	Test en laboratorio nodos en circuito impreso	2 días	sáb 07/04/18	lun 09/04/18
28	**Realización software colector V1**	**11 días**	**mar 10/04/18**	**sáb 21/04/18**
29	Parte comunicación nodo colector	7 días	mar 10/04/18	mar 17/04/18
30	Parte tratamiento de datos	4 días	mié 18/04/18	sáb 21/04/18
31	**Aquisición de material para completar nodos**	**4 días**	**lun 23/04/18**	**jue 26/04/18**
32	Material nodos finales	4 días	lun 23/04/18	jue 26/04/18
33	Material desarrollo en pcb nodos finales	4 días	lun 23/04/18	jue 26/04/18
34	**Diseño y fabricación resto circuitos**	**5 días**	**vie 27/04/18**	**mié 02/05/18**
35	Diseño/Modific. placas mediante software PCBs	2 días	vie 27/04/18	sáb 28/04/18
36	Fabricación placas pcb finales	5 días	vie 27/04/18	mié 02/05/18
37	**Pruebas en laboratorio del conjunto**	**2 días**	**jue 03/05/18**	**vie 04/05/18**
38	Test en laboratorio del conjunto	2 días	jue 03/05/18	vie 04/05/18
39	Mejoras software colector	2 días	jue 03/05/18	vie 04/05/18
40	Mejoras nodos (calibración, estabilidad, etc)	2 días	jue 03/05/18	vie 04/05/18
41	**Despliegue de nodos**	**14 días**	**sáb 05/05/18**	**dom 20/05/18**
42	Despliegue nodos interior D080	3 días	sáb 05/05/18	mar 08/05/18
43	Validación de datos recibidos	1 día	mié 09/05/18	mié 09/05/18
44	Posibles correciones/mejoras (Interior)	2 días	jue 10/05/18	vie 11/05/18
45	Despliegue nodos exterior	3 días	sáb 12/05/18	mar 15/05/18
46	Validación de datos recibidos	1 día	mié 16/05/18	mié 16/05/18
47	Posibles correciones/mejoras (Exterior)	4 días	jue 17/05/18	dom 20/05/18
48	**Comienzo PEC4 (Memoria TFM)**	**19 días**	**lun 21/05/18**	**dom 10/06/18**
49	Elaboración memoria TFM	19 días	lun 21/05/18	dom 10/06/18
50	**Comienzo PEC5 (Defensa TFM)**	**11 días**	**lun 11/06/18**	**vie 22/06/18**
51	**Preparación defensa TFM**	**11 días**	**lun 11/06/18**	**vie 22/06/18**
52	Elaboración presentación	11 días	lun 11/06/18	vie 22/06/18
53	Respuestas a preguntas planteadas	11 días	lun 11/06/18	vie 22/06/18

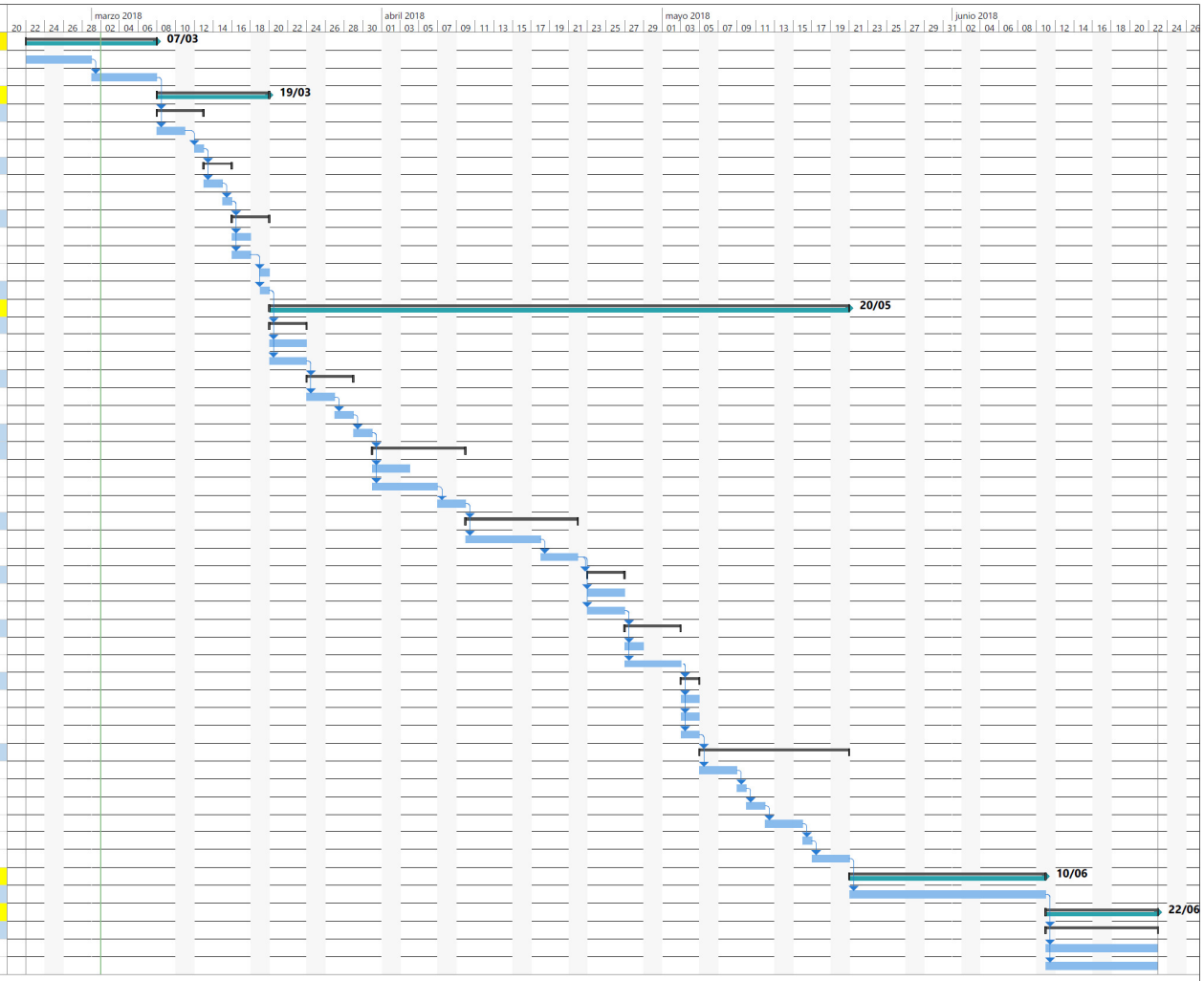


Ilustración 9. Planificación del trabajo del TFM en 5 fases principales que principalmente vienen condicionadas por las fechas de entrega de las pruebas de evaluación

- Fase 1. Definición TFM: En esta fase se definen el alcance, los objetivos del presente trabajo y los pasos a seguir para su consecución.
- Fase 2. Estudio trabajos relacionados: Durante este periodo se realiza un análisis profundo del estado del arte del mercado, abarcando dos aspectos principales, por un lado, el estudio de cómo se mide el *seeing* en otros observatorios y el estado actual y evolución de las redes de sensores inalámbricos.
- Fase 3. Producto del TFM: En esta fase se llevará a cabo la realización de los productos en sí mimos, es decir, tanto del hardware como del código software necesario para el correcto funcionamiento del conjunto.
- Fase 4. Memoria TFM: Las tareas dentro de esta fase están focalizadas a la depuración y acabado de la memoria del presente proyecto.
- Fase 5. Defensa TFM: Esta etapa está dedicada a la elaboración de la documentación necesaria para poder presentar y defender correctamente todos los hitos, decisiones, productos, etc. Derivados del presente trabajo.

1.5. Breve sumario de productos obtenidos

A continuación, se comentan brevemente todos los productos que se han obtenido en la elaboración de este TFM.

- Red de sensores compuesta por 10 nodos para recolectar valores de temperatura:
  - 9 nodos para sensado (cada nodo recoge 4 valores de temperatura).
  - 1 nodo colector.
- Diseño del circuito electrónico y placa PCB de los nodos de sensado.
- Fabricación de los circuitos de los nodos sensores, por tanto, la creación de las PCBs y la integración de todos sus componentes.
- Diseño y fabricación de cajas para albergar la electrónica que compone cada nodo sensor mediante impresora 3D.
- Software para la configuración de los nodos desarrollada en Python 3.
- Software para la recolección de datos desarrollada en Python 3.
- Software para la recolección de datos de calidad de imagen desde las bases de datos del OAJ.
- Sistema de monitorización de los datos obtenidos mediante el uso de aplicaciones de código abierto.
- Manuales de instalación y de usuario.

1.6. Breve descripción de los otros capítulos de la memoria

Los capítulos en los que se ha dividido la presente memoria son:

- **Estado del Arte:** En este capítulo y las subsecciones que lo conforman se ha tratado de realizar una revisión histórica de cómo ha sido estudiado el *seeing* local en otros observatorios a lo largo de los últimos años en todo el mundo. También se hace una breve referencia al uso de las redes

de sensores en los observatorios astrofísicos profesionales y en las aportaciones que este proyecto realiza en dicho campo.

- **Fundamentos de las redes de sensores:** Se ha tratado de dar una visión general de qué son las WSNs, las características de los elementos que las conforman y las consideraciones a tener en cuenta en su diseño.

- **Muestra de algunas de las tecnologías actuales en las WSNs:** Se ha recopilado información de las características principales de algunas de las tecnologías más utilizadas en la actualidad para las redes de sensores y se ha realizado una comparativa entre las mismas. Finalmente se ha hecho una justificación de la tecnología elegida.

- **Descripción de los elementos utilizados:** En estas secciones se describen, de forma detallada, cada uno de los elementos utilizados para la consecución de los objetivos fijados, se trata de una descripción tanto de los elementos hardware como software.

- **Sistema prototipo:** Se hace una revisión de los prototipos obtenidos y su despliegue en laboratorio, es decir, sin realizar todavía su instalación en un entorno tan complejo y exigente como es el observatorio.

- **Fase de despliegue:** A lo largo de este capítulo se recoge todo el proceso de instalación y monitorización realizada en el propio OAJ.

- **Valoración económica del trabajo:** En este apartado se recoge el coste económico de la elaboración de los productos derivados de este TFM.

- **Análisis de los resultados:** Mostramos aquí los resultados obtenidos en la implementación final de la red de sensores en el observatorio.

- **Conclusiones:** En este capítulo se desglosarán las lecciones aprendidas, los aciertos y los puntos de mejora, así como acciones futuras recomendadas.

- **Bibliografía:** Se muestra un listado de toda la documentación, e información consultada para la elaboración de este TFM.

- **Anexos:** Se incluyen aquí documentos variados relacionados con el presente proyecto, algunos como planos, códigos, esquemas, etc.

2. Estado del Arte

A lo largo de los siguientes apartados se pretende realizar un breve resumen, por orden cronológico, de cómo se ha medido el *seeing* local en otros observatorios astrofísicos y mostrar cómo el presente proyecto, introduce aspectos novedosos mediante la utilización de las WSNs para este cometido.

2.1. Antecedentes

Tal y como se ha indicado, podemos describir el *seeing* local como la degradación en la calidad óptica de la imagen que se introduce debido a la turbulencia del aire en la vecindad de los telescopios. También se ha indicado que existen varias fuentes de *seeing* local, aunque los elementos que contribuyen en mayor medida a esta degradación son la cúpula, entendida como todas las superficies del cerramiento en general (se estudian todas las superficies incluido el suelo en los alrededores del telescopio) y los espejos.

Hasta hace unos años, a estas contribuciones se las conocía como un todo, y se les daba el nombre de *seeing* de cúpula, así, a mediados de los años 70, aunque sus efectos eran ya conocidos, los estudios sobre este fenómeno eran puramente anecdóticos, en gran medida por la dificultad que existía para diseñarlos e interpretar los resultados [18].
Sin embargo, en esos años se empieza a hacer el comisionado de grandes telescopios, clase 4m (el término clase aquí hace referencia al tamaño del espejo primario, así por ejemplo el tamaño del espejo del telescopio principal del OAJ es de 2,5 m de diámetro, mientras que por otra parte el gran telescopio de canarias [19] siendo el mayor telescopio óptico del mundo cuenta con un espejo primario de 10 metros de diámetro) y el fenómeno del *seeing* de cúpula empieza a llamar la atención de los astrónomos cuando se dan cuenta de que las observaciones hechas con esos grandes telescopios y ópticas mejoradas, no eran capaces de obtener la calidad y resolución esperadas, así pues, aunque la causa general del *seeing* de cúpula fue rápidamente reconocida como generada por la turbulencia que se crea en el interior de las mismas debido a efectos de convección del aire, el fenómeno en sí, era, simplemente, objeto de conjeturas y unos pocos experimentos muy distantes en el tiempo.
Así pues, en las décadas sucesivas, sobre todo con la proliferación de las pruebas y el aumento de datos recogidos, se empieza a estudiar el fenómeno y poco a poco se empieza a entender y mejorar las instalaciones para mitigar sus efectos. A continuación, se mostrarán de forma cronológica alguno de los ejemplos más relevantes en el estudio del *seeing* local.

2.2. Medida del seeing en otros observatorios

A continuación, se muestra un breve resumen, ordenado cronológicamente, de los métodos usados en distintos observatorios para la medición del *seeing* local.

2.2.1. Telescopio Corporación Canadá-Francia-Hawái (1991)

Ilustración 10. Observatorio CFHT en Hawái (Mauna Kea) [18]

El observatorio Canadá, Francia, Hawái (CFHT) es un observatorio astronómico que está instalado en la cima del monte Mauna Kea en la isla de Hawái, está situado a una altitud de 4024msnm y en la actualidad alberga un telescopio óptico e infrarrojo de clase 3.6m [20].

En el año 1991, Derrick Salmo, David Cowley y Jerry Sovka, publicaron un estudio [21] en el que se evaluaba como las diferencias de temperatura entre el espejo primario y el aire ambiente, y entre el aire en el interior y en el exterior de la cúpula afectaban a la degradación en la calidad de imagen.
En su estudio, instalaron desde el año 1986 en numerosas localizaciones alrededor del telescopio y la cúpula, termistores que eran directamente conectados a un *data logger* que recogía datos cada 10 minutos almacenándolos en un disco. Además de estos datos, también se registraba la velocidad del viento. De esta forma, y con la ayuda de una cámara de alta resolución, conocida como HRCam que era capaz de tomar gran cantidad de imágenes, fue posible correlacionar la calidad de dichas imágenes con las diferencias de temperatura.
Las conclusiones a las que llegaron fueron que el *seeing* de espejo produce un empeoramiento en la imagen de 0.40”/ºC$^{6/5}$ (Para trata de entender estas medidas hay que analizar en detalle el informe del CFHT [20] , se trata de un tema complejo pero básicamente estas unidades provienen del ángulo de dispersión de la turbulencia debido al modelo de Kolmogorov. En el citado documento se puede apreciar como en las ecuaciones (2) y (5) las diferentes contribuciones del *seeing* son sumadas en potencias de 5/3. Esta contribución de 5/3 viene de la integral de la ecuación (1) que a su vez viene con un exponente 3/5, por tanto elevando los dos lados de la ecuación se obtiene el 5/3. A modo de resumen se puede concluir que el hecho de elevar a 5/3 viene de la teoría de la turbulencia de Kolmogorov originada por la pendiente que tiene la energía de los elementos turbulentos en función de su tamaño, dicha pendiente es lo que se conoce como rango inercial en el que los elementos turbulentos mayores transfieren, a modo de cascada, su energía a elementos menores, hasta que estos son tan pequeños que la energía se disipa por viscosidad) si el espejo primario está más caliente que el aire en el interior de la cúpula y que el *seeing* de cúpula contribuye en una ratio de 0.1”/ºC$^{6/5}$ de la diferencia de temperatura entre el aire interior y el exterior de la cúpula.

2.2.2. Observatorio Gemini (1993)

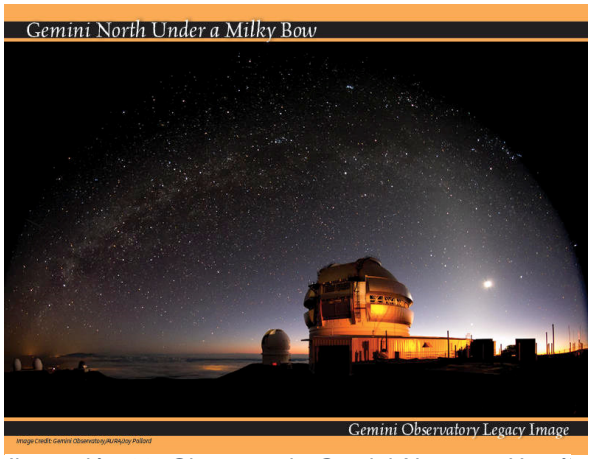


Ilustración 11. Observatorio Gemini Norte en Hawái (Mauna Kea) [20]

Gemini es un observatorio astronómico conformado por una cooperación internacional de países (Estados Unidos, Canadá, Brasil, Argentina y Chile) dispone de dos telescopios gemelos ópticos/infrarrojos de 8.1m de apertura y que están ubicados en distintos hemisferios del planeta, uno de ellos, conocido como Gemini Norte, está situado en el monte Mauna Kea en Hawái a una altitud de 4213msnm mientras que el otro, Gemini Sur, está ubicado en la cumbre del Cerro Pachón en Chile, a una altitud de 2722msnm *[22]*.

En el año 1993, la oficina de proyecto del Gemini publicó un estudio en que el presentaba una estrategia para el control del *seeing* producido por la cúpula de su instalación en Mauna Kea [23], para ello crearon un modelo térmico de 130 elementos, dicho modelo les permitía simular la radiación térmica del cielo nocturno, la de la tierra y el ratio de flujo volumétrico del aire ambiente a través del interior de la cúpula. Debido a que compartían ubicación, para el modelo se tomaron los datos de temperatura y velocidad de viento obtenidos del estudio del CFHT. El software utilizado fue Ideas-TMG [24] que se basa en una representación y cálculo por diferencias finitas. Los resultados obtenidos se plasmaron en este estudio, y se trata de una completísima estrategia para el control de *seeing*, recomendaciones del *coating* de las cúpulas (ahora es bien conocido que el exterior de las cúpulas debe ser color plata si se trata de un telescopio nocturno y blanco si es solar gracias a estudios como este) y formulación con la que calcular la contribución al *seeing* según la diferencia de temperaturas entre las superficies de la cúpula y el aire ambiente.

2.2.3. Observatorio Astrofísico de Cagliari (1998)

El Observatorio Astronómico Cagliari de INAF-OAC es una instalación de investigación que forma parte del Instituto Nacional de Astrofísica italiano (INAF) dedicado al estudio del Universo y los elementos que lo componen. El sector de investigación que más caracteriza a INAF-OAC es la radioastronomía, sobre todo gracias a que gestiona el Radio Telescopio de Cerdeña (SRT), un radiotelescopio grande y nuevo construido en el municipio de San Basilio [25].

Ilustración 12. Observatorio de Cágliari [23]

En el año 1993 F.Buffa, G.C. Cocco, I. Porceddu y M. Serrau presentaron un estudio [26] en el que mostraban cómo los gradientes de temperatura entre los espejos del telescopio Galileo y el aire que los rodea eran la fuente primaria de

seeing local y como, con una correcta termalización de la cúpula durante el día, se podían mitigar sus efectos. Lo que proponían era que, con una correcta previsión meteorológica se podrían prefijar las consignas de temperatura para el sistema de climatización de la cúpula, para ello se instaló una estación meteorológica en una torre de 15m, con la intención de ubicarla a la misma altura del espejo primario, a la que iban conectados 4 sensores de temperatura a distintas alturas, se trataba de pt100 estándar con una precisión de 0.1ºC, un sensor de humedad, un barómetro y un sensor de velocidad y dirección del viento de tipo ultrasónico. El sistema adquiría datos cada 30 segundos y eran enviados al edificio del telescopio mediante fibra óptica. Para la predicción de la temperatura se realizaba un post procesado de los datos y mediante la utilización de filtros matemáticos (filtros Kalman y procedimientos de redes neuronales) se realizaba la predicción meteorológica.

2.2.4. Real Observatorio Greenwich (1998)

El Real Observatorio de Greenwich [27] es uno de los observatorios más antiguos del mundo (1675) se encuentra ubicado en la ciudad de Greenwich, Londres – Inglaterra, y se considera como punto de origen para los meridianos. Debido a la contaminación y a la iluminación nocturna se trasladó al castillo de Herstmonceux, si bien en la actualidad, su actividad se centra en tareas de investigación y divulgación.

Ilustración 13. Real Observatorio de Greenwich [25]

En el año 1998, M.T. Bridgeland, C.R. Jenkins, presentaron un trabajo [28] en el que estimaban el *seeing* de espejo en un laboratorio, se trataba de un método no intrusivo y por tanto se podía utilizar en el propio telescopio, la idea se basaba, tal y como se realiza en otros estudios, en la medición de las variaciones de temperaturas que son las que producen turbulencias. Para dicha medición se utilizaron termopares, termómetros de resistencia de alambre delgado y termistores de micro-perlas (*micro bead thermistors*) y se estudió la variación de temperaturas entre pares de sensores. Los resultados obtenidos trataron de ser verificados mediante técnicas ópticas, concretamente mediante la utilización de un láser de helio-neón y un interferómetro, y se demostró que las variaciones que se daban en el patrón de las franjas de interferencia eran consistentes con los datos obtenidos por los termistores y con la ley de turbulencia de los fluidos de Kolmogorov.

2.2.5. Observatorio del Roque de los Muchachos. Telescopio William Herschel (1999)

Ilustración 14. Telescopio William Herschel en la isla de La Palma (Islas Canarias) [17]

El telescopio William Herschel es un telescopio reflector de 4'2 metros de diámetro, forma parte del grupo de telescopios Isaac Newton ubicados en el observatorio del Roque de Los Muchachos en la isla de La Palma en las Islas Canarias a una altitud de 2344msnm [19].

En el año 1999, R. W. Wilson, N. O'Mahony, C. Packham y M. Azzaro presentaron un estudio sobre el seeing en el telescopio William Herschel. Para ello, utilizaron dos instrumentos, el primero de ellos, un monitor de seeing de tipo DIMM situado en las cercanías de la cúpula que alberga dicho telescopio, con él se medía el *seeing* intrínseco del emplazamiento, mientras que el segundo era un monitor de *seeing* basado en un sensor de frente de ondas *Shack-Hartmann* [29] (apodado JOSE, del inglés *Joint Observatories Seeing Evaluation project*) que instalado en el propio telescopio, ya que está dotado de un foco Nasmuth que mediante un tercer espejo proporciona otro punto focal en uno de los lados del tubo del telescopio, permitía obtener valores de *seeing* en el interior de la cúpula, de esta forma se pudieron comparar las medidas tomadas por ambos monitores.
Tras el estudio concluyeron que la contribución al *seeing* por parte de la cúpula no era significativa.

2.2.6. Observatorio Astronómico Nacional de San Pedro Mártir (2007)

Ilustración 15. Observatorio San Pedro Mártir ubicado en Baja California (México) [28]

Está situado en la sierra de San Pedro Mártir en Baja California, México y es operado por el Instituto de Astronomía de la Universidad Autónoma de México (UNAM). Se trata del observatorio más importante de México, y el situado a mayor altura del país, 2,830 msnm. Actualmente cuenta con tres telescopios cuyos diámetros son 2.1 m, 1.5m y 0.84 m [30].

A principios del año 2007, una serie de autores presentaron un trabajo [31] en el que medían la contribución de la capa superficial en los alrededores del observatorio desde los 2.3m hasta 15m de altura al *seeing* óptico. Para ello utilizaron un mástil con sensores de temperatura micro-diferenciales localizados a 7 diferentes alturas, un par de sensores a cada una de las alturas separados mediante una varilla, 0.95m entre sí y 2 pares en el punto más elevado, que tomaban muestras cada 5ms, con ello,

debido a la diferencia de temperaturas obtenidas por cada par de sensores, se conseguía medir el índice de refracción o perfil C_n^2 en los primeros 15 metros. Por otro lado, se utilizó un monitor DIMM durante 23 noches para determinar el *seeing* intrínseco del emplazamiento, determinando una mediana de *seeing* de 0".84 y siendo el *seeing* promedio debido a la capa superficial de 0".16. Así pues, determinaron que la turbulencia óptica de esta capa tiene una contribución promedio del 5.2% del C_n^2 total lo que corresponde a una degradación promedio del *seeing* del 3.2%, llegado a la conclusión de que la presencia de árboles en el sitio no influye de manera considerable en el *seeing* debido a la capa superficial.

2.2.7. Telescopio Discovery Channel (2007)

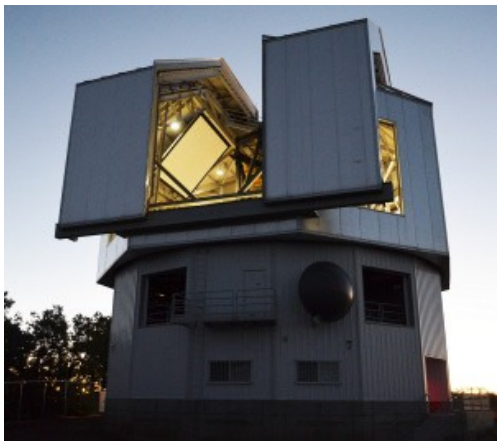


Ilustración 16. Telescopio Discovery Channel ubicado en el bosque Coconino (Arizona-USA) [30]

El telescopio Discovery Channel (DCT) es un telescopio de 4.3 m del Observatorio de Lowell y situado en el bosque natural de Coconino en Arizona, Estados Unidos. Se encuentra a una altitud de 2360msnm y fue un regalo del CEO de *Discovery Channel*, John Hendricks, al observatorio. Está valorado en 16 millones de dólares americanos, y a cambio, *Discovery Channel* puso nombre al telescopio y tiene los derechos de uso de las imágenes en sus programas y documentales [32].

En el año 2007, Dan Blanco presenta un trabajo sobre la estimación del *seeing* local en el DCT [11], para ello se basará en los trabajos realizados por Robert Ford en el Gemini y René Racine en el CFHT, que han sido citados en los apartados anteriores, en los que se estudiaba la contribución al *seeing* local de los distintos elementos y se proponía una formulación sencilla para medir dicha contribución en base a diferencias de temperatura.

Para su estudio, Dan Blanco, recurrió a software de simulación térmica, concretamente al programa *ThermoAnalytics* [16], en el que se puede establecer la ubicación del observatorio y el programa determina la radiación solar en función de la fecha, hora, etc. Después una vez introducido el modelo de la cúpula y demás elementos e introduciendo algunas estimaciones como la velocidad de viento en ciertos puntos, mediante el análisis de elementos finitos se pueden obtener unos resultados muy fiables de la evolución térmica de la instalación, por tanto, es posible predecir la variación térmica entre diferentes superficies y, por consiguiente, realizar una predicción de la evolución del *seeing* de cúpula a lo largo de todo el periodo simulado.

La decisión más destacada de este estudio es que se cambió el color blanco con el que iba a ser pintada la cúpula a un color plata. También se estimó que el mejor *seeing* se obtiene en las primeras horas de la noche cuando las temperaturas están lo más próximas al equilibrio con la temperatura del aire ambiente, por otra parte, se llegó a la sorprendente conclusión de que la diferencia en la velocidad del viento utilizada, el modelo se corrió utilizando dos

velocidades de viento diferentes, 4.76m/s y 2.1m/s, apenas tenía impacto en el *seeing* de cúpula.

2.2.8. Instituto Astrofísico de Canarias. Instrumento g-SCIDAR (2008)

El instituto de Astrofísica de Canarias (IAC) es un centro de investigación localizado en las Islas Canarias. Se trata de un consorcio público que está integrado por la Administración del Estado Español, el Gobierno Canario, la Universidad de La Laguna y el Consejo Superior de Investigaciones Científicas (CSIC) en el que además participan instituciones de 19 países. Está ubicado en el municipio de San Cristóbal de La Laguna en la Isla de Tenerife y cuenta con dos observatorios en los que hay multitud de telescopios e instrumentos gestionados por los distintos países. Por un lado, está el Observatorio del Roque de Los Muchachos a 2396 msnm, dónde destaca el mayor telescopio óptico del mundo, el Gran Telescopio de Canarias (GTC o GRANTECAN) y por el otro está el Observatorio del Teide situado en la zona de Izaña a 2371 msnm [5].

Ilustración 17. Observatorio del Roque de los Muchachos (IAC) ubicado en Tenerife [33]

Aunque la técnica, en su versión clásica, SCIDAR, fue propuesta en el año 1974, todavía hoy es un proyecto vivo en el que se están realizando continuos estudios. En el año 2008, J.J. Fuensalida, B. García-Lorenzo y C. Hoegemann [34] presentan un trabajo sobre la corrección de la contribución del *seeing* de cúpula mediante los datos recogidos por el instrumento g-SCIDAR mediante el análisis de Fourier.

El g-SCIDAR es un instrumento que mide perfiles de turbulencia atmosférica mediante la observación de estrellas binarias, esto le permite obtener la información de localización en altura de las capas de turbulencia, el movimiento de estas capas turbulentas y, por tanto, las velocidades de las mismas.

Como se observa, a priori, el instrumento obtiene la estructura vertical de la turbulencia atmosférica, es decir el perfil C_n^2, y en este trabajo se presenta la técnica basada en el análisis de Fourier para extraer de este perfil la contribución de la cúpula y el espejo al *seeing*.

Aunque la formulación de esta solución es compleja y queda fuera del alcance de este trabajo, a modo de resumen se puede decir que, el procedimiento se basa en la observación de estrellas binarias y la utilización de cinco correlaciones

cruzadas consecutivas de los *frames* con centelleo (debido al *seeing*) y se fundamenta en el reconocimiento de la forma de la función de autocorrelación de la turbulencia de la cúpula en cada correlación cruzada.

2.2.9. Gran Telescopio Binocular (2010)

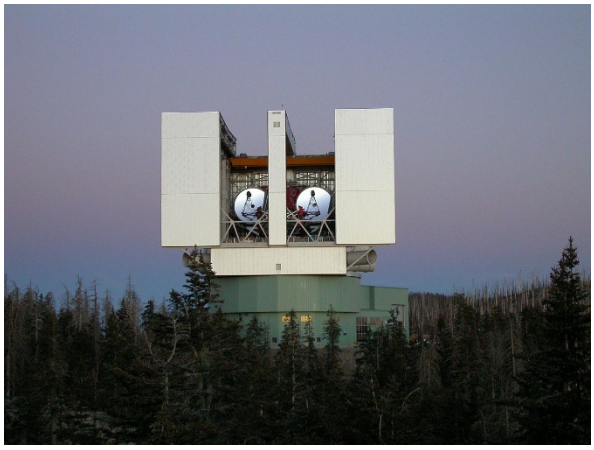


Ilustración 18. Gran Telescopio Binocular ubicado en el monte Graham (Arizona – USA) [33]

El gran telescopio binocular (LBT del inglés *Large Binocular Telescope*) es un proyecto gestionado por un gran consorcio de diferentes instituciones y países, situado en el Monte Graham en Arizona a una altitud de 3200msnm, está formado por dos espejos de 8.4m montados sobre una base común. Se trata de uno de los telescopios más avanzados tecnológicamente y con mayor resolución del mundo debido al uso de técnicas de óptica adaptativa [35].

En el año 2010, E. Masciadri, J. Stoesz, S. Hageli y F. Lascaux, presentan un trabajo [36] sobre como caracterizar la distribución vertical de la turbulencia atmosférica, es decir, el perfil C_n^2, todos los parámetros que se derivan de él y los perfiles de viento para el LBT. Para ello hacen uso durante 43 noches del instrumento g-SCIDAR y al igual que en la propuesta del IAC se estudian estrellas binarias que mediante cálculos de autocorrelación permiten crear mapas de centelleo. En estos mapas se forman tres picos llamados tripletas en donde los picos posteriores son proporcionales a la fuerza de la turbulencia de las capas locales (entre 200-250m y 20-30m de altura), mediante técnicas de correlación son capaces de estimar la contribución de la cúpula llegando a la conclusión que el valor medio de contribución de *seeing* de dicha cúpula es de 0.52 arcsec, observándose además, una fuerte dependencia con la estacionalidad, por lo que se recomienda una investigación más detallada.

2.2.10. Instituto Astronómico Estatal Shtérnberg (2011)

El Instituto Astronómico Estatal Shtérnberg es el mayor centro de formación y de estudios astronómicos de Rusia. Sus actividades se extienden a casi todas las esferas de la astronomía moderna y de la gravimetría. Al mismo tiempo, es el centro de operación de los astrónomos que estudian la física de los sistemas binarios y múltiples, la estructura, la cinemática y la dinámica de nuestra galaxia y galaxias del grupo local, recibiendo el apoyo de las principales escuelas de investigación de Rusia [37].

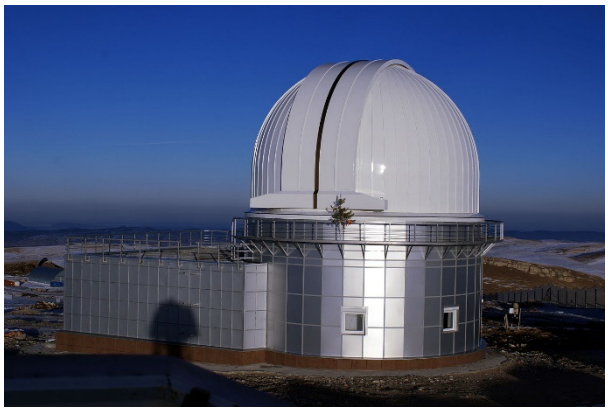


Ilustración 19. Observatorio de las montañas del Cáucaso [35]

De este instituto dependen varios centros astronómicos entre ellos el Observatorio de las Montañas del Cáucaso con un telescopio reflector de 2,5m de diámetro sobre el que en el año 2009 S.A. Potanin presenta un trabajo [38], en el que describe un método, basado en un detector de frente de ondas Shack-Harman, para determinar la calidad óptica de dicho telescopio.

En el 2011 este mismo autor, presenta un método [39] basado en los conocimientos adquiridos en su trabajo anterior, para la estimación del seeing de cúpula de los telescopios AZT-22 de 1.5m y Zeiss de 1.0 m del Observatorio de Maidanak en Uzbekistan y del telescopio ZTSh del observatorio de Crimea en Ucrania, basándose también en el uso de un sensor de frente de onda Shack-Hartman.
La idea que hay detrás de este tipo de sensores es que los rayos divergentes provenientes de las estrellas se hacen pasar a través de un cubo divisor del haz, estos rayos debidamente colimados atraviesan un conjunto de lentes que hacen que cada rayo enfoque en un CCD diferente y en función de la variabilidad de estos rayos se puede inferir el *seeing*.
Así pues, aunque la teoría óptica detrás de estos sensores no es sencilla, a modo de resumen, podemos decir que la técnica empleada es similar a la de los dispositivos DIMM. Mediante el sensor Shack-Hartman se obtienen las inclinaciones locales del frente de onda en distintas sub-aperturas. Al disponer de un conjunto de bases fijas de distinta longitud se puede estudiar la dispersión en las diferencias de inclinación en función de dicha longitud y de allí deducir el *seeing* de cúpula.

2.2.11. Domo Fuji en la Meseta Antártica (2013)

Ilustración 20. Telescopio DIMM en la torre de 9m en Domo Fuji (Antártida) [39]

Se espera que la Antártida sea uno de los mejores lugares para observaciones astronómicas terrestres. Debido a la baja temperatura, el ruido térmico en las longitudes de onda infrarrojas es mucho más bajo en la Antártida que en otros sitios templados. La atmósfera seca con poco vapor de agua es más transparente en infrarrojo a sub-milimétrico. En las cumbres de la meseta, la velocidad del viento es baja y la atmósfera es estable, por lo que no existen tormentas violentas ni tormentas de nieve. En ese contexto, Japón ha organizado un consorcio formado por cuatro universidades (Tohoku, Tsukuba, Rikkyo, Nagoya) y dos

institutos (Instituto Nacional de Investigación Polar y Observatorio Astronómico Nacional) para promover la astronomía en Domo Fuji [40] .

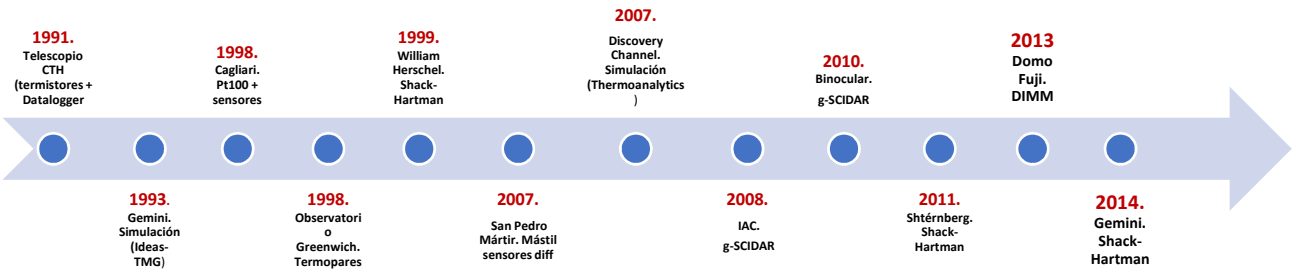
Aunque las características de esta estación son absolutamente diferentes al no existir cúpula como tal, sí que cabe nombrar el estudio [41] que diferentes autores presentaron sobre la medición del *seeing* atmosférico y la estimación de la contribución de altura de los instrumentos sobre el nivel de la superficie nevada a dicho *seeing*.
La técnica empleada se basa en el uso de monitor diferencial de imagen (DIMM) en el rango visible (472 nm). En este estudio se llegó a la conclusión de que el *seeing*, tal y como se esperaba, es excelente llegándose a valores por debajo de 0.2" y que son típicos valores de *seeing* de 0.3" de forma continuada. Por otra parte, se determina que la altura límite para conseguir estos valores es de 11m.

2.2.12. Observatorio Gemini (2014)

En el año 2014 en el Observatorio Gemini Sur, del que ya hemos hablado anteriormente, varios autores presentan un trabajo para medir el *seeing* [42] basándose en la correlación cruzada espaciotemporal de las muestras obtenidas por cinco sensores de frente de ondas Shack-Hartman con lo que se analiza la turbulencia atmosférica a diferentes altitudes. El método proporciona información útil para la operación de los sistemas de óptica adaptativa, como el número de capas de turbulencia, sus velocidades, altitudes y fuerza, y también un mecanismo para estimar la contribución del *seeing* de cúpula a la turbulencia total.

2.2.13. Línea temporal de la evolución en el estudio del seeing local

A continuación, se muestra una breve línea temporal en la que se visualizan cronológicamente los estudios sobre el seeing local anteriormente comentados.



2.2.14. Resumen

Como vemos, en la literatura consultada, existen dos grupos de aproximaciones para tratar de cuantificar la contribución de la cúpula a la degradación del *seeing* atmosférico.

Uno de estos grupos se basa en la utilización de instrumentación óptica para estudiar diferentes comportamientos de la luz e inferir de ellos el *seeing* local.
Otro grupo se basa en la medida de la variación de fenómenos ambientales, sobre todo variaciones de temperatura, bien sea de forma real o simulada, y tratar de correlacionar dichas variaciones con las medidas de *seeing* atmosférico. El objetivo de este proyecto entraría pues dentro de este último grupo.

2.3. Uso de las WSNs en los observatorios astrofísicos.

A continuación, se muestra brevemente cómo surgieron y han ido evolucionando las redes de sensores y el nivel de implantación de las mimas en los observatorios astrofísicos profesionales.

2.3.1. Breve descripción de la evolución histórica de las WSNs

Para poder entender las WSN actuales conviene hacer un breve resumen a su historia y evolución. Las primeras redes de sensores que podríamos considerar como tal, tienen un origen militar, se trataba del Sistema de Vigilancia Sónica (SOSUS, del inglés *Sound Surveillance System*) que consistía en una red de boyas submarinas desplegadas por el ejército de los Estados Unidos con el fin de detectar submarinos soviéticos. Su desarrollo se inició en 1949 y su puesta en marcha en 1961. (La primera detección de un submarino soviético ocurrió en 1962) [43].
Haciéndose eco de las inversiones realizadas en la década de los 70s y 80s para el desarrollo del hardware de la red ARPANET la agencia militar de investigación avanzada de Estados Unidos, comenzó en el año 1980 el proyecto *Distributed Sensor Networks* (DSN) en el que poco a poco fueron participando universidades como la *Carnegie Mello University* y *el Massachusetts Institute of Technology Lincoln Labs*, con lo que las WSNs encontraron su nicho en las investigaciones académicas y civiles.
Si bien, en esta primera fase, ciertas limitaciones como el costo de implementación, la falta de estándares de red, el consumo, la escalabilidad impedían el despliegue generalizado en una gama amplia de aplicaciones.
No obstante, aunque la aplicación a gran escala en aplicaciones industriales y domésticas no era posible, sí que, a finales del siglo XX y principios del XXI, tanto en el ámbito académico como en el industrial se reconoce el amplio potencial de las redes de sensores, por lo que se aúnan esfuerzos dando lugar a diferentes iniciativas como:

- *UCLA Wireless Inegrated Network Sensors* (1993).
- *University of California at Berkeley PicoRadio Program* (1999).
- *μ Adaptative Multi-domain Power Aware Sensors Program* MIT (2000).
- *NASA Sensor Webs* (2001).
- *ZigBee Alliance* (2002).
- *Center for Embedded Network Sensing* (2002).

El objetivo de todas ellas es reducir los costes de despliegue e incrementar su funcionalidad aplicando los avances tecnológicos. La culminación de este esfuerzo se da con la llegada del Internet de las Cosas (IoT).

2.3.2. Aplicaciones basadas en WSNs en observatorios astrofísicos

A pesar de que las redes de sensores inalámbricas gozan ya de un cierto grado de madurez y de que son evidentes las ventajas que presentan, aunque se hablará sobre ello más adelante, a modo de resumen cabe destacar su versatilidad para el sensado de fenómenos físicos, con gran rapidez de despliegue, escalabilidad y bajo coste, resulta sorprendente la escasa o prácticamente nula, literatura en la que se describa la utilización de este tipo de sistemas en observatorios astronómicos.
Entre todos los trabajos consultados, destacar el estudio [44] presentado en 2012 por los autores Tongying Li y Zhenchao Zhang en el que se reconoce la importancia que los cambios ambientales ejercen en el rendimiento de los telescopios y como, la utilización de redes inalámbricas de sensores para la medición en tiempo real de dichos cambios, puede ayudar a los sistemas de control a tomar las medidas necesarias para mitigar sus efectos.
Finalmente indicar, que, aunque el uso de WSNs sea practicante nulo o la literatura sobre su aplicación meramente anecdótica en observatorios astrofísicos, no ocurre lo mismo en otros ámbitos de investigación, siendo frecuente la utilización de los mismos en campos tan diversos como la medicina, oceanografía, vulcanología, meteorología, y un largo etc.

2.4. Aportaciones del presente proyecto

Este proyecto, tal y como se ha dicho, pretende evaluar la viabilidad de la utilización de las redes de sensores en el estudio de la degradación del seeing debido a las variaciones térmicas locales.
Por otra parte, tal y como se ha comentado, la utilización de esta tecnología en los observatorios astrofísicos es puramente anecdótica, y este trabajo es, por tanto, un estudio que podríamos considerar casi pionero en la utilización de los mismos en estos entornos de trabajo.
Desde el punto de vista de este autor, el mundo de las WSNs abre un amplio abanico en las diferentes ramas que convergen en un observatorio astrofísico, dónde no sólo el campo de la física está presente, sino que también existe una contribución vital de diferentes ramas de la ingeniería.
No obstante, su aplicación y concretamente en lo referente a este proyecto, no está exenta de retos a superar, retos que serán tratados en profundidad en los sucesivos capítulos de este trabajo.

Así pues, con el fin de recapitular, podemos decir que las aportaciones más importantes que realiza este trabajo son, en primer lugar la de ser una punta de lanza en la introducción de las redes de sensores en los observatorios astrofísicos, en segundo, realizar un estudio inicial de las diferentes tecnologías y su encaje dentro de una instalación de este tipo, y para finalizar, en tercer lugar, la de proponer un método comparativamente muy flexible, rápido y barato para medir las contribuciones a la degradación del seeing de diferentes elementos que prestan servicio en la operación habitual de los telescopios.

3. Fundamentos de las redes de sensores

A continuación, se mostrará una breve pincelada de las características más relevantes que, en un sentido amplio, toda red de sensores debería poseer, así como los elementos que las componen.

3.1. Introducción a las redes de sensores

Una red de sensores, idealmente, está compuesta por dispositivos de pequeñas dimensiones (al menos tan reducidas como sea posible), bajo coste y consumo de energía que se denominan nodos, estos nodos se emplean para la monitorización de ciertas variables dentro de un entorno (ya sean ambientales, de eficiencia energética, industriales, automoción, médicas, domóticas, y un largo etc.) Y por ello se despliegan dentro del fenómeno del que se desean realizar las mediciones. Algunas otras características que los definen son, capacidad de trabajo sin supervisión, la adaptabilidad al ambiente y el hecho de no necesitar la definición previa su posición. Esto permite, por ejemplo, su rápido despliegue de una forma aleatoria en zonas con difícil acceso o en operaciones de socorro y salvamento. Por otra parte, esto también significa que los protocolos de las redes de sensores y sus algoritmos debe tener capacidades auto-organizativas, a lo que se le suele añadir la capacidad de realizar cooperación entre nodos para las tareas de enrutamiento y transferencia de la información. Normalmente estos nodos están equipados con un pequeño procesador que les permite realizar tareas simples de computo sobre los datos, conversiones de señales analógicas a digitales (ADC, del inglés *Analog to Digital Converter*) y actuación sobre entradas y salidas. En las redes de sensores, además de los nodos existe un elemento clave que es colector, cuya misión consiste, por una parte, en la recolección de todos los datos y por otra en la de servir de interfase con otras redes, ya sea una red local, Internet, etc. [45].

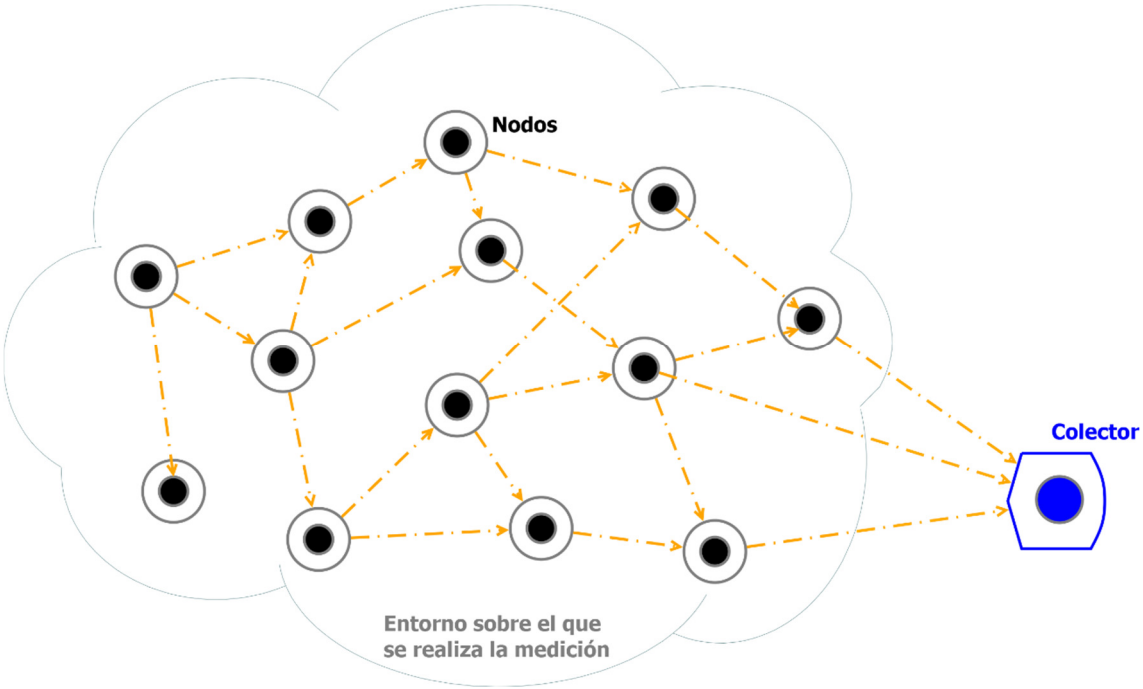


Ilustración 21. Esquema tipo de una red de sensores, se observa que, al menos, existe un nodo colector y una serie de nodos finales para realizar la medición

3.2. Arquitectura de un nodo sensor

Podemos dividir la arquitectura de un nodo en cuatro bloques principales [45], la unidad de sensado o detección, la unidad de procesado, un transmisor y la unidad de alimentación.

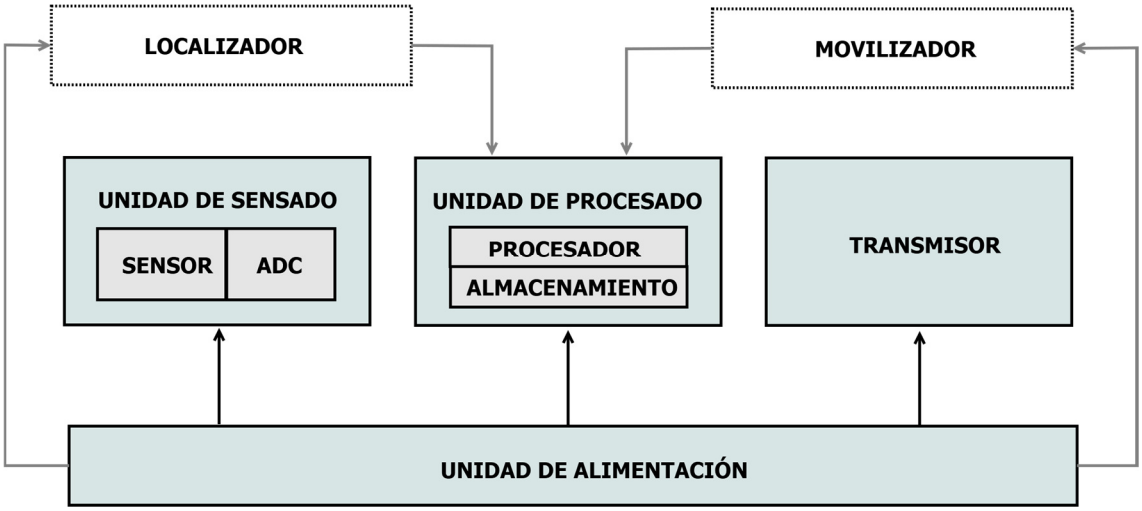


Ilustración 22. Arquitectura fundamental de un nodo WSN, no obstante, otros módulos podrían ser añadidos en función de la aplicación a realizar

Como podemos ver en la imagen anterior, la unidad de sensado normalmente está compuesta por dos sub-unidades, los sensores propiamente dichos y los conversores analógicos-digitales (ADC), que permiten convertir la señal desde el mundo analógico valores adecuados para su procesamiento en el mundo digital.
Por su parte la unidad de procesado normalmente está formada por un pequeño microprocesador y una pequeña sub-unidad de almacenamiento, este bloque se encarga principalmente de gestionar los procedimientos de colaboración entre nodos y el acondicionamiento de los datos a transmitir. La función del transmisor es la de garantizar los procedimientos de transmisión y recepción inalámbrica que permitirán al nodo el envío y recepción de la información con los diferentes nodos dentro de la red.
Finalmente encontramos la unidad de alimentación para proporcionar potencia al resto de elementos, un correcto diseño de esta unidad es vital para alargar la vida útil y el máximo aprovechamiento del sensor dentro de la red sin la necesidad de recargas o cambios de la propia unidad de alimentación.
Además de estos bloques, los nodos sensores pueden incorporar otras unidades que serán dependientes de la aplicación, alguna podría ser la unidad de localización necesaria para algunas técnicas de enrutamiento y sensado y de movilidad, cuando es necesario mover el nodo para realizar ciertas tareas [45].

3.3. Factores a tener en cuenta en el diseño de una red de sensores

Existen diversos factores a considerar a la hora de diseñar una red de sensores, será importante tenerlos en consideración para poder elegir la tecnología que mejor se adapte a las necesidades de nuestra aplicación [45].

- Tolerancia a fallos: Los sensores pueden fallar por bloqueos, pérdidas de energía, daños ambientales, interferencias, etc. El fallo de algún nodo sensor no debería afectar al funcionamiento global de la red, es por ello por lo que cuando hablamos de tolerancia a fallos en las redes de sensores, nos referimos a la capacidad de dichas redes de seguir operando ante pérdidas de operatividad de alguno de sus nodos.
- Escalabilidad: El número de sensores desplegados en el entorno a estudiar puede ir desde unos pocos nodos hasta miles de ellos. Las redes de sensores deben permitir añadir y eliminar nodos según las necesidades de la tarea a realizar.
- Limitaciones hardware: Tal y como se ha mostrado los sensores están formados básicamente por cuatro unidades, todas estas unidades deben tener un tamaño pequeño que posibiliten su despliegue en diversos ambientes.
- Topología: Son diversas las topologías que se pueden utilizar en una red de sensores, pero básicamente podemos indicar que la elección de una de ellas dependerá de las capacidades de los nodos, la cantidad de los mismos a utilizar y de los requerimientos de nuestra aplicación. (Topologías en estrella, malla, híbridas, etc.)
- Tecnología de transmisión: Aunque se hará un estudio más detallado en posteriores capítulos, podemos decir que son diversas las tecnologías de transmisión inalámbrica que se pueden utilizar, Bluetooth, Zigbee, WiFi, etc. Y la elección de cada una de ellas estará determinada por los requerimientos de nuestra aplicación y las condiciones de entorno.
- Consumo de energía: Dado que los nodos en las redes de sensores sólo pueden ser equipados con una fuente de potencia limitada, el máximo aprovechamiento de la misma es fundamental, más si cabe, dado que en algunos escenarios el reemplazo de estas fuentes de energía es prácticamente imposible.

4. Muestra de algunas tecnologías en las WSN actuales

Los procesos de estandarización en el campo de las WSN han estado muy activos en los últimos años y existe una amplia variedad de tecnologías a emplear cuya elección dependerá del objetivo de la aplicación. En este apartado se hará un breve resumen de algunas de las tecnologías actuales más relevantes

4.1. Estándar IEEE 802.15.4

Se trata de un protocolo estándar para comunicaciones sin hilos, creado por el IEEE (del inglés, *Institute for Electrical and Electronics Engineers*) [46] y que está destinado para ser usado en redes de bajo coste, corto rango de alcance, baja potencia y baja tasa de transferencia. El estándar especifica las dos capas más bajas de la pila de protocolos, concretamente la capa física (PHY) la capa de control de acceso al medio (MAC), las capas superiores de la pila y las de

interoperabilidad son definidas separadamente por otros estándares como 6LoWPAN, ZigBee, ISA100.11a y *WirelessHART* [47].
El estándar define dos tipos de nodos los FFD (del inglés *Full-Function Devices*) y RFD (del inglés *Reduced-Function Devices*), los primeros pueden actuar como coordinadores y comunicarse con otros FFDs o RFDs, mientras que los segundos actúan como dispositivos finales y sólo se pueden comunicar con los FFDs. Las topologías, a priori soportadas son punto a punto y estrella.
Soporta en las bandas de frecuencia de 868/915 MHz unas tasas de transferencia de 100/250 Kbps, y en la banda de 2,4 GHz con una tasa de trasferencia de 250 Kbps.
En su origen, esta especificación marcó cuatro capas físicas, que utilizan modulaciones de espectro ensanchado (tanto DSSS como PSSS) [48], en la actualidad, se ha hecho una modificación a este estándar, es el conocido IEEE 802.15.4a, que añade dos capas físicas más que utilizan la banda ultra-ancha (UWP) y técnicas de espectro ensanchado (CSS) [49].

4.1.1. Zigbee:

Como se ha dicho es el protocolo usado en las capas superiores de la pila de protocolos, que se apoyan en el 802.15.4 Este estándar es mantenido por la *ZibBee Aliance* [50] y su principal contribución es la de dotar a las aplicaciones de la topología de malla, esto es muy útil para redes donde el nodo colector, llamado coordinador en esta especificación, está más allá del rango del nodo sensor, puesto que ciertos dispositivos intermedios que reciben el nombre de *routers*, son capaces de reenviar el mensaje.
Zigbee ha sido diseñado para las aplicaciones de bajo consumo, por tanto, se adecúa para los sistemas embebidos [51] y aquellos mercados donde la disponibilidad y la versatilidad se priorizan sobre los anchos de banda.
Finalmente indicar que Zigbee utiliza 16 canales en la banda 2,4GHz hasta 2,48Ghz. Algunas de sus características resumidas son:

- **Estándar:** ZigBee 3.0 basado en IEEE 802.15.4.
- **Frecuencias de trabajo:** 902-928 MHz (América y Australia), 868-868.6 MHz (Europa) y no licenciadas 2.4-2.4835 GHz (Mundial).
- ***Bitrate***: 250Kbps.
- **Alcance**: Hasta 100m a 2.4GHz, varias decenas de km a otras frecuencias.
- **Seguridad:** AES-128.
- **Campo de aplicación:** Principalmente, automatización, monitorización, y domótica.

4.1.2. 6LoWPAN:

Definido por IETF (del inglés, Internet *Engineering Task Force*) [52] para adaptar IPV6 al IEEE 802.15.4, así estos dispositivos se pueden comunicar directamente con otros dispositivos IP.
Debido a que en IPv6 el tamaño de los paquetes es mayor al *frame* de 802.15.4 existe una capa de adaptación para la comprensión de la cabecera y la fragmentación y re-ensamblaje de la trama.

- **Estándar:** 6LoWPAN (RFC6282) basado en IEEE 802.15.4.
- **Frecuencias de trabajo:** 902-928 MHz (América y Australia), 868-868.6 MHz (Europa) y no licenciadas 2.4-2.4835 GHz (Mundial).
- ***Bitrate***: Desde 20-250Kbps.
- **Alcance**: Típicamente 10-20m.
- **Seguridad:** AES-128.
- **Campo de aplicación:** Industria, monitorización y domótica.

4.1.3. WirelessHART

Lanzado en el 2007 es un estándar destinado a las aplicaciones industriales, añade la posibilidad de utilizar el protocolo HART [53] y por tanto hacerlo compatible con dispositivos HART, Se basa en la existencia de dispositivos *gateway* que le sirven de conexión entre los dispositivos de campo y los *hosts* de la planta. Existe un equipo administrador de red para la configuración, manejo de rutas y monitores de todos los elementos.

- **Estándar:** WirelessHART (HART 7.0) basado en IEEE 802.15.4.
- **Frecuencias de trabajo:** 902-928 MHz (América y Australia), 868-868.6 MHz (Europa) y no licenciadas 2.4-2.4835 GHz (Mundial).
- ***Bitrate***: Hasta 250Kbps.
- **Alcance**: Hasta 60m.
- **Seguridad:** AES-128.
- **Campo de aplicación:** Principalmente destinado a la industria relacionado con procesos de monitorización.

4.1.4. ISA100.11a

Desarrollado por el comité de estandarización ISA100 (del inglés, *International Society of Automation*) [54] su mayor campo de aplicación es la industria de la automatización. Está compuesta por dispositivos de campo que realizan las tareas de sensado y algunos de ellos también realizan tareas de ruteo, por otra parte, existen dispositivos *gateways* para comunicar la WSN con las aplicaciones de usuario. Presentan una baja latencia y un rápido tiempo de respuesta (100ms), usa la banda de 2.4GHZ con saltos en frecuencia para prevenir las interferencias.

- **Estándar:** ANSO/ISA100.11a.
- **Frecuencias de trabajo:** 2.4 GHz.
- ***Bitrate***: Hasta 250Kbps.
- **Alcance**: Hasta 500 m.
- **Seguridad:** AES-128.
- **Campo de aplicación:** Principalmente control de procesos industriales, servicios de localización y logística.

4.2. Bluetooth y Bluetooth Low Energy (BLE)

Se basa en la tecnología bluetooth, por tanto, opera en la banda de 2.45 GHZ. Debido a la libre operación de esta banda, los sistemas deben soportar la interferencia que produce el medio. Para reducir la interferencia producida, Bluetooth utiliza la técnica de Espectro Ensanchado por Saltos de Frecuencia (FHSS, del inglés *Frequency-Hopping Spread Spectrum* [55]). Su capacidad de transmisión es más alta (hasta 3Mbps) pero con un rango corto de alcance (10-500m). La nueva especificación de Bluetooth introduce la versión de bajo consumo (BLE) que permite a los dispositivos trabajar con baterías durante meses o años, además, esta nueva especificación, soporta la topología en malla para establecer comunicaciones muchos-a-muchos. (Esta topología es soportada con la especificación *core* 4.0 y superiores) Es muy apropiado para sistemas sanitarios, deportes, seguridad y entretenimiento [56].

- **Estándar:** IEE 802.15.1 (Especificación Bluetooth 5.0).
- **Frecuencias de trabajo:** 2.4 GHz.
- ***Bitrate***: Hasta 3Mbps.
- **Alcance**: Hasta 500 m.
- **Seguridad:** 56-128 bit *key* y AES-128.
- **Campo de aplicación:** Aunque es posible su aplicación en distintos campos, la más extendida es en los electrodomésticos de consumo, la última especificación está centrada en el IoT.

4.3. Z-Wave

Es un estándar de baja potencia basado en comunicaciones en radio frecuencia, fue diseñado por Zensys [57] para el control remoto de aplicaciones residenciales e iluminación de zonas comerciales, en la actualidad es gestionado por Z-Wave Alliance [58]. Trabaja en las bandas de 900MHz y en la de 2.4GHz con ratios de transmisión de 9.6Kbps y 100Kbps respectivamente, un rango máximo de 100m.

- **Estándar:** Z-Wave Alliance ZAD12837 / ITU-T G.9959.
- **Frecuencias de trabajo:** 900MHz y 2.4 GHz(ISM).
- ***Bitrate***: Desde 9.6Kbps hasta 100Kbps.
- **Alcance**: 100 m.
- **Seguridad:** AES-128.
- **Campo de aplicación:** Domótica, seguridad y electrodomésticos de consumo.

4.4. ANT

Es una tecnología propietaria para aplicaciones inalámbricas de ultra-bajo consumo. Trabaja en la banda de 2.4GHz y soporta diversas topologías como punto a punto, estrella, árbol y malla. Tiene una tasa de transferencia muy alta, si la comparamos con 802.15.4, de hasta 1 Mbps. Su principal problema es la interoperabilidad, aunque para solucionarlo se ha desarrollado el protocolo

ANT+, soportado por la ANT+ Alliance [59] se trata de una tecnología muy usado en dispositivos deportivos de monitorización.

- **Estándar:** Propietaria.
- **Frecuencias de trabajo:** 2'4GHz.
- ***Bitrate***: Hasta 1 Mbps.
- **Alcance**: Hasta 30m.
- **Seguridad:** AES-128.
- **Campo de aplicación:** Dispositivos de monitorización deportiva, como pulsómetros, podómetros, etc.

4.5. LoRa/LoRaWAN

Son protocolos de comunicación inalámbrica [60] diseñados específicamente para el Internet de las cosas (IoT, del inglés Internet of Things). Tienen un bajo ancho de banda, 250bits-50Kbits y un alcance de hasta 20km en campo abierto, con un bajo consumo de energía.
Mientras LoRa ha creado su propio ecosistema de antenas y concentradores LoRa, de forma que los usuarios pueden registrar sus sensores Lora en él, y así acceder a los datos transmitidos por los sensores en los servidores Lora de la nube (exige una cuota), LoRaWAN permite la creación de nuestra propia red, en este caso, los sensores Lora se registran y transmiten información a concentradores y *gateways* propios y son estos últimos lo que vía conexión Ethernet/WiFi/3G/etc. Envían los datos recibidos a nuestros servidores propios en la nube.

- **Estándar:** LoRaWAN siguiendo el estándar IEEE 802.15.3.
- **Frecuencias de trabajo:** 433, 868, 915 MHz.
- ***Bitrate***: Hasta 50kbps.
- **Alcance**: hasta 20Km (ambientes rurales).
- **Seguridad:** AES-128.
- **Campo de aplicación:** Principalmente IoT, también se usa en iluminación inteligente, sensores inteligentes, seguimiento y localización de paquetes, domótica, monitorización de constantes vitales, agricultura, etc.

4.6. Wavenis

Al igual que LoRa, Wavenis [61] es un protocolo de comunicación inalámbrica diseñado para el Internet de las Cosas. Permite un largo alcance, aunque algo menor que LoRa, 5Kms, sin embargo, se pueden usar repetidores para mejorar el alcance.
Su ancho de banda es sensiblemente superior al de Lora que puede alcanzar los 250KBits, además también hace hincapié en el bajo consumo de energía y permite crear nuestra propia red local.

- **Estándar:** Propietario
- **Frecuencias de trabajo:** 868/915 MHz
- ***Bitrate***: Desde 10 Kbps hasta 250 kbps

- **Alcance**: 200m interior – 5Km exterior.
- **Seguridad:** AES-128.
- **Campo de aplicación:** Principalmente IoT, domótica, seguimiento de paquete.

4.7. SigFox

SigFox [62] también ha sido diseñado para el Internet de las Cosas. Permite un alcance de hasta 50Km dentro del paraguas de cobertura de SigFox y bajo consumo de energía, sin embargo, su ancho de banda es muy limitado, envía paquetes de 12 bytes a 600 baudios. Se trata de una red propietaria, lo que implica el pago por cada equipo conectado sin posibilitar la creación de redes locales.

- **Estándar:** Propietario.
- **Frecuencias de trabajo:** Europa 868 MHz, América del Norte 902 MHz, resto 920 MHz.
- ***Bitrate***: 100 – 600 bps. Depende del contrato, pero el máximo son 140 mensajes de subida de 12 bytes y 4 mensajes de bajada de 8 bytes.
- **Alcance**: Dependiente del proveedor (Típico 50 km rural, 10 km urbano)
- **Seguridad:** AES-128.
- **Campo de aplicación:** Principalmente IoT y monitorización.

4.8. Dash7

Es una tecnología open *source*, de ultra-bajo consumo y largo alcance, hasta 5 Km, pensada especialmente para las redes de sensores. Puede operar en las bandas de 433MHz, 868MHz y 915MHz. La vida de las baterías puede durar años con tasa de transferencia de casi 200 Kbps. En la actualidad es soportado por la DASH7 *Alliance Protocol* (D7A) [63].

- **Estándar:** DASH7 Alliance basado en ISO/IEC 18000-7.
- **Frecuencias de trabajo:** 433/868/915MHz**.**
- ***Bitrate***: Hasta 166.667 kbps.
- **Alcance**: hasta 5 Km.
- **Seguridad:** AES-128.
- **Campo de aplicación:** Sistemas de monitorización y control, como por ejemplo control de parkings, detectores de presencia, seguimiento de ítems, etc.

4.9. EnOcean

Es una tecnología emergente, utilizada para domótica y automatización industrial usa las bandas de 315/868 MHz con unos rangos de alcance de 30m para interior y 300m para exterior. La tecnología EnOcean se basa en la utilización energética eficiente, por ello son frecuentes los módulos que funcionan sin batería ya que

aprovechan la energía que se genera a su alrededor de distintas formas (pulsación de un botón, apertura de una puerta, solar, etc.). En la actualidad el protocolo es gestionado por la EnOcean Alliance [64].

- **Estándar:** EnOcean Wireless estándar (ISO/ IEC 14543-3-1X).
- **Frecuencias de trabajo:** 868, 902, 928 MHz.
- ***Bitrate***: 125 kbps.
- **Alcance**: Hasta 300 m.
- **Seguridad:** AES-128.
- **Campo de aplicación:** Principalmente domótica, control de iluminación, aplicaciones máquina-a-máquina (M2M del inglés, *Machine to Machine)*.

4.10. RFID Activos

El funcionamiento de RFID se basa en la utilización de unos dispositivos llamados tarjetas o tags, en los cuales se puede almacenar y recuperar información. Un tag RFID se basa en la generación de una señal de radiofrecuencia que contiene los datos almacenados y que suelen ser datos de identificación.
De una forma general, podemos decir que el sistema RFID consta de tres componentes: los tags o transpondedores, los lectores o transceptores y el subsistema de procesamiento de datos. Existen tags pasivos que no necesitan ninguna fuente de alimentación ya que la señal que llega a los propios tags induce una corriente eléctrica pequeña pero suficiente para operar el circuito integrado de la propia tarjeta, etiquetas activas que poseen su propia fuente de energía con lo que son capaces de transmitir señales con mucho mayor alcance (rangos de varios cientos de metros con una vida útil de hasta 10 años es posible) y etiquetas semi-pasivas que también incorporan una fuente de alimentación pero su misión es la de alimentar al microcontrolador de la tarjeta y no la de transmitir la señal.

No hay ninguna corporación pública mundial que gestione las frecuencias utilizadas para RFID, así que cada país fija sus propias reglas, no obstante, en Europa, por ejemplo, existen ciertas administraciones como ERO, CEPT, ETSI que regulan las comunicaciones, aunque tienen que ser ratificadas por las administraciones nacionales [65].
A pesar de ello, a nivel general podemos decir que existen etiquetas RFID de baja frecuencia (LF: 125 - 134 kHz y 140 - 148.5 kHz) y de alta frecuencia (HF: 13.56 MHz) se pueden utilizar de forma global sin necesidad de licencia. La frecuencia ultra alta (UHF: 868 - 928 MHz) no puede ser utilizada de forma global, ya que no hay un único estándar global [65].

4.11. Comparativa entre tecnologías

	IEEE 802.15.4	IEEE UWB 802.15.4a	Bluetooth	BLE	Z-Wave	ANT	LORA	Wavenis	SigFox	Dash7	EnOcean
Frecuencia	868/915 MHz; 2.4 GHz	3.1-10.6 GHz	2.4 GHz	2.4 GHz	Sub-1GHz y 2'4 GHz	2.4 GHz	433, 868, 915 MHz	868, 915 MHz	868/902/920 MHz	433/ 868/915MHz	868, 902, 928 MHz
Max througput	250 kbps	110Mbps	3 Mbps	1 Mbps (En foros hablan de 250kbps)	100 kbps	1 Mbps	50 kbps	250 kbps	100-600 bps	166'667 kbps	125 kbps
Rango	100 m-100Km	100 m	10-500 m	200m	100 m	30 m	20 km	20m-5 km	10-50 km	5 km	300 m
Batería	Días/Años	Años	Años	Meses/Años	Años	Año	Años	Años	Años	Años	Sin batería
Topología	Estrella, P2P, Malla	Estrella, P2P, Malla	P2P	P2P (malla en la nueva espec.)	Malla	Estrella, P2P, árbol, Malla	Estrella, P2P, árbol, Malla	P2P	Estrella	Estrella, árbol, mala	Malla
Consumo	Bajo	Bajo	Bajo	Ultra Bajo	Bajo	Ultra Bajo	Ultra bajo	Ultra Bajo	Ultra Bajo	Bajo	Ultra Bajo
Abierta	Si	Si	Si	Si	Si	No	Si	Si	No	Si	Si
IPv6	Si	No	No	Si	Si	No	Si	Si	Si	No	No
Mercado	Sensado, mediciones, dispositivos inteligentes en malla	Monitor en tiempo real y posicionamiento interiores, telemedicina	Electrodomésticos de consumo	Dispositivos inteligentes, hogar, salud	Domótica, seguridad, electrodomésticos de consumo	Salud, deporte, ritmo cardiaco, sensores de velocidad	IoT, monitorización y control de procesos industriales	Sensado, mediciones, IoT, telemetría, domótica	IoT, sistemas de monitorización	Monitorización y sistemas de control	Domótica, automatización industrial, M2M
Facilidad Starter Kits	Sí (Varios para Zigbee)	Sí (precio elevado en comparación)	Sí	Si (pero sin implementar malla)	Si (Kits de domótica)	No	Si	No	Sí (precio elevado en comparación	Sí (precio elevado en comparación)	Sí

4.12. Tecnología elegida. Justificación

A lo largo del presente capítulo se va a tratar de justificar porque, tras el análisis de las soluciones actuales, se ha optado por definir la tecnología Zigbee como la más apropiada para la consecución de los objetivos propuestos para este proyecto.

4.12.1. Razones tecnológicas

Durante los últimos años, la ZigBee Alliance ha hecho importantes mejoras en el protocolo, haciendo esta tecnología muy simple, robusta, flexible, tolerante a fallos y con un precio muy ajustado, además su bajo consumo energético, confiere a los nodos la capacidad de trabajar con pequeñas baterías durante largos periodos de tiempo [66].
Por todo ello, ZigBee es un protocolo inalámbrico ampliamente extendido en aplicaciones de monitorización y control, permitiendo además una topología en malla, lo cual se considera muy importante para la realización de este proyecto, pues otorga un rango de alcance extremadamente largo con el simple hecho de ir añadiendo nodos con la capacidad de repetir la señal, es decir, *routers*.
La especificación de ZigBee está disponible de forma gratuita, lo que permite profundizar en el conocimiento del protocolo lo que también es ideal dentro de un marco académico como este.
Otra ventaja muy importante de ZigBee es que permite a los nodos trabajar con modos de suspensión o reposo (*sleep modes*) lo que reduce el consumo y alarga la vida de las baterías. Además, los módulos ZigBee pueden pasar del estado de suspensión al estado activo en menos de 15 ms, por tanto, pueden ser utilizados en dispositivos que exijan baja latencia, esta característica también ha sido tenida muy en cuenta para este proyecto, dado que, en otras tecnologías, como por ejemplo Bluetooth, el paso del estado de reposo a activo suele introducir retardos considerables, del orden de unos pocos segundos, lo que hubiese restado versatilidad en los modos de muestreo de las temperaturas, obligándonos además a tener muy presente esta demora.

4.12.2. Razones prácticas

A la hora de buscar en el mercado kits de iniciación (starter kits) se ha encontrado que o bien ciertas tecnologías no brindan esta oportunidad, o su precio es mucho más elevado que el de los kits ZigBee sin prestar ninguna característica imprescindible o vital para este trabajo.
Así por ejemplo, aunque los dispositivos Bluetooth están ampliamente extendidos, para este proyecto se quería utilizar una topología en malla que confiriese un rango de alcance prácticamente ilimitado, y aunque, como se ha dicho, la última especificación de Bluetooth admite dicha topología es extremadamente complicado encontrar módulos que permitan esta configuración, en la actualidad sólo se han encontrado dispositivos aceptables en la casa *Nordic Semiconductors* [67], no obstante su precio es mucho más elevado que el de los nodos ZigBee sin aportar ventajas significativas.
Por el contrario la casa Digi dispone de unos nodos llamados XBee que permiten trabajar fácilmente con este protocolo [68], se trata de módulos muy asequibles que se pueden adquirir por menos de 20€/módulo (más tarifas de importación dependientes de la cantidad a comprar), además dichos módulos incorporan la

posibilidad de leer entradas analógicas y digitales sin la necesidad de añadir un microcontrolador adicional lo cual encaja a la perfección con el objeto de este TFM.
Finalmente indicar que existe una amplia comunidad que utiliza estos dispositivos lo que ha originado un foro muy activo al que se puede acceder desde la propia web de Digi y que proporciona, además, un eficaz servicio de soporte al cliente, dispone de multitud de documentación libre, ejemplos, casos de uso y no menos importante, proporciona de manera gratuita, librerías Java, Python o C para poder ser utilizadas para desarrollar nuestra propia aplicación

A modo de conclusión se indican aspectos concluyentes de algunos protocolos que aconsejan o desaconsejan su uso para este proyecto:

- Bluetooth: Quedó descartada principalmente por su corto alcance a efectos prácticos, ya que, si se quiere incrementar dicho alcance, se debe utilizar la tecnología en malla, y en la actualidad no existen dispositivos en el mercado que ofrezcan las herramientas necesarias para un proyecto en un entorno y condiciones como el actual.

- Z-Wave: Su orientación está fuertemente dirigida hacia aplicaciones domóticas lo que no encaja exactamente con la finalidad este proyecto.

- ANT: Se desestima por ser una tecnología propietaria.

- Wavenis: También se desestimó por ser una tecnología pensada para el IoT y topologías P2P.

- UWB 802.15.4a: Dotado de grandes anchos de banda lo que encarece el precio sin estar justificado para esta aplicación.

- SigFox: Es propietario y tiene una tasa muy baja de transferencia por lo que queda descartado.

- Dash7, EnOcean, LoraWan y RFID activos: Se consideran tecnologías muy interesantes y que podrían encajar en este y otros proyectos en el observatorio, no obstante, su precio es muy elevado en comparación con los módulos XBee y las ventajas que aportan no justifican ese desembolso extra ya que no se consideran necesarias para este trabajo.

Se concluye por tanto que la mejor solución para este proyecto es utilizar el protocolo ZigBee y contar con los módulos de XBee de Digi que se detallarán en capítulos posteriores.

5. Descripción de los elementos utilizados

Una vez que se ha determinado la utilización de la tecnología ZigBee para la realización de este proyecto se procede a hacer una revisión más detallada de los distintos elementos que conforman el producto final, tanto a nivel hardware como software.

5.1. Hardware

A continuación, se describirán todos los elementos hardware que conforman el dispositivo final, tratando no sólo los componentes en sí mismos, sino que también se hablará de cómo se ha realizado la fabricación de los productos finales.

5.1.1 Sensores inalámbricos (WSN)

Tal y como se ha indicado, se ha optado por la utilización de los módulos XBee de la casa Digi Inc, que pasaremos a describir en mayor detalle a continuación.

5.1.1.1. Tipos de módulos XBee (ZigBee)

Digi ofrece en la actualidad tres tipos de módulos XBee para la implementación de comunicaciones entre sensores ZigBee (Evidentemente nos hemos centrado en el protocolo elegido 802.15.4 y ZigBee, no obstante, Digi ofrece módulos XBee que trabajan con diferentes tecnologías, 802.15.4, DigiMesh, WiFi. 900MHz, 868MHz, LTE, IoT y 3G).

- Serie 1: Se trata de la versión más sencilla de los XBee ya que sólo admiten la conexión punto a punto siendo su principal campo de utilización la sustitución de las comunicaciones serie cableadas.

- Serie 2: Es una versión algo más compleja que la anterior ya que proporciona la posibilidad de utilizar una topología en malla que describiremos a continuación, además permite utilizar tanto ZigBee como el protocolo DigiMesh, este último propietario y desarrollado por Digi.

- Serie 3: Se trata de la última versión de los módulos XBee e introduce algunas mejoras sobre la versión anterior, una de ellas es la posibilidad de configurar distintas tecnologías en el mismo módulo, entre ellas la versión de ZigBee 3, aumenta la potencia de transmisión en algún canal y disminuye la potencia de operación.

Además, todas estas series cuentan con su versión PRO cuya única diferencia a nivel funcional es que aumentan el rango de alcance, evidentemente esto se consigue a costa de un aumento también en el consumo energético y por supuesto se traduce en un incremento en el precio de los módulos.

5.1.1.2. Versión elegida

Lógicamente la Serie 1 queda descartada al tener únicamente la capacidad ser usadas en topologías punto-a-punto, por lo que se barajan la Serie 2 y 3, esta última no tiene un precio ostensiblemente mayor que la anterior y a priori podría parecer la más indicada dada su versatilidad. Lamentablemente en la actualidad Digi sólo ofrece para la Serie 3 dos versiones de este módulo una versión para montaje superficial (SMT, del inglés *Surface Mount*) y otra micro (MMT, de inglés *Micro Mount*) que no sólo dificultan la elaboración de las placas PCB compatibles con este tipo de encapsulados, sino que hacen imposible la elaboración de prototipos en laboratorio fáciles de ser montados y modificados.
A diferencia de la Serie 3 la Serie 2 ofrece una versión con pines aptos para ser soldados en una PCB (TH, del inglés *Trough Hole*) o ser adaptados para su conexión a una *breadboard* y sus características siguen encajando perfectamente en el objetivo de este proyecto, por tanto, se define la Serie 2 de Digi XBee como la ideal para la realización de nuestros dispositivos.

Una vez que se decide utilizar la Serie 2 se estudia si será su versión estándar o la versión PRO la elegida, la estándar proporciona un rango máximo de 60m en el interior y de 1,2Km en el exterior, mientras que la PRO va desde 90m en el interior hasta 3,2Km en el exterior. En este proyecto este aumento del alcance no está justificado, ya que con el margen actual es más que suficiente y dada la posibilidad de utilizar topologías en mallas tampoco se prevé como una fuente de problemas.

La última decisión que se debe tomar es que tipo de antena se va a utilizar ya que los módulos elegidos presentan cuatro posibilidades:

- Antena integrada en la propia PCB del módulo.
- Conector U.FL [69].
- Conector RPSMA [70].
- Con antena tipo cable integrada.

De las opciones anteriores a priori se descartan las versiones con conector por dos motivos fundamentales, el primero es que son ostensiblemente más caras ya que se necesita adquirir de forma separada la antena, sin que este incremento en el precio sea justificado dado que las necesidades del proyecto no prevén desplazar la antena lejos del nodo en sí mismo y el segundo es que dado que se pretende integrar la electrónica en una caja estanca, el tener este puerto de conexión abierto podría ser fuente de problemas dadas las duras condiciones meteorológicas del Pico del Buitre.
Dentro de las dos opciones restantes se considera la opción de la antena integrada en la PCB como la más robusta ya que en la basada en cable, aunque posibilitaría la opción de sacar la antena al exterior (daría más versatilidad sin incrementar el precio) al ser dicha antena más accesible, es más propensa a roturas y averías.
Finalmente indicar que también se han hecho pruebas de alcance con el módulo con la antena integrada en la PCB insertando dicho módulo en cajas estancas provisionales y se ha tratado de dificultar la emisión/recepción por diversos medios, y en todos los casos se ha obtenido unos buenos resultados y unos valores más que aceptables en la potencia de la señal (este valor se puede determinar desde el propio módulo con el software que proporciona el fabricante) por tanto es la versión Digi XBee

S2C TH con antena integrada en la PCB la considerada óptima para el proyecto, el *parth number* del fabricante para estos módulos es : XB24CZ7PIT-004.

Ilustración 23. Módulo Digi XBee elegido. Se trata de la versión S2C-TH con antena integrada en la PCB [68]

5.1.1.3. Entradas analógicas y conversor AD

Tal y como se ha comentado en capítulos anteriores, una de las características que hacen más atractiva la utilización de estos módulos para este proyecto es que dichos módulos incorporan la posibilidad de utilizar cuatro entradas analógicas (numeradas desde ADC0 hasta ADC3) sin la necesidad de incorporar ningún controlador adicional. El valor máximo de entrada en voltaje es de 1.2v, siendo 3.3v la alimentación normal del módulo.
Cada una de estas entradas analógicas está dotada de su propio conversor AD de 10 bits de resolución, por tanto, como se verá más adelante, se propone la conexión de un sensor de temperatura analógico, correctamente adaptado, a cada una de estas entradas (Se incluye el esquema eléctrico en el anexo *"12.1 Esquema eléctrico"*).

5.1.1.4. Digi Mesh Kit

Digi ha puesto en el mercado unos kits de iniciación a un precio razonable (en capítulos posteriores se hará una valoración económica de todos los elementos) con los que es posible realizar pruebas iniciales de comunicación entre módulos y valorar las limitaciones y puntos fuertes de esta tecnología. Además de los propios módulos se incluyen una serie de herramientas necesarias para su configuración y testeo.
La versión del Kit adquirida es la que cumple con las características anteriormente expuestas, cuyos elementos se muestran a continuación:

- Parth-Number: XKB2-Z7T-WZM.
- Elementos que incorpora:
  - Tres módulos XBee S2C, dos de ellos con empaquetamiento TH y otro SMT, este último se utilizará como nodo colector.
  - Tres tarjetas de desarrollo, estas tarjetas permiten la conexión de los XBee a un ordenador para poder acceder a ellos vía puerto serie, además incorpora conectores para introducir señales analógicas/digitales/PWM [71], etc.
  - Tres cables micro-USB para realizar la conexión al PC.

- Dos pegatinas con el logo Digi-XBee.

Ilustración 24. Elementos que integran el Digi Zbee Zigbee Mesh Kit y que ha sido adquirido para la realización de las pruebas iniciales dentro de este TFM [68]

5.1.1.5. Parametrización de los nodos XBee

Los módulos XBee admiten la lectura y/o modificación de una serie de parámetros que afectan directamente a su modo de funcionamiento [72], aunque con el software desarrollado para este proyecto todos ellos serán accesibles y se podrán modificar los que sean de escritura, tanto vía local como remota (vía RF), aunque para esta aplicación, sólo unos pocos de ellos exigirán la modificación de los valores que presentan en su configuración por defecto.
Se trata en general de los parámetros más relevantes por lo que a continuación se detallarán y se justificará su modificación.

a) *ID. PAN ID* (Identificador de red)

Este registro admite valores hexadecimales en un rango de:

0 - 0xFFFFFFFFFFFFFFFF.

Se utiliza para indicar al nodo a qué red pertenece de forma que establezca comunicación sólo con los nodos que pertenecen a su red. Si introducimos el valor 0 el nodo se puede comunicar con cualquier red sin embargo un valor 0 en el nodo coordinador (colector) hace que este seleccione aleatoriamente un valor.
Para este proyecto se ha elegido el valor aleatorio de 2015, no obstante, puede ser modificado libremente, eso sí, teniendo en cuenta que debe modificarse vía local porque de lo contrario se pierde la comunicación con dicho nodo al establecer el cambio.

b) *JV. Channel Verification* (Verificación del canal)

Admite los valores de:

0 – Deshabilitado.
1 – Habilitado.

Si está habilitado los nodos verifican la existencia de un nodo coordinador en la red para asegurarse de que están trabajando en un canal correcto, en caso de no estarlo dejará el canal para buscar en otro. Si este registro no está habilitado siempre permanecen en el mismo canal. En este proyecto se ha fijado a 1 en todos los nodos.

c) CE. Coordinator Enable (Fijar como coordinador)

Configura el nodo como nodo coordinador de la red.

0 – Deshabilitado.
1 – Habilitado.

Si está habilitado el nodo realizará el papel de coordinador y colector de datos. Tal y como se ha indicado, será el nodo que se conecte al equipo de control (Pc y/o raspberry) usando la tarjeta proporcionada por el Mesh Kit.
Dando una visión más general de la red, el protocolo ZigBee admite la existencia de tres elementos fundamentales:

- Nodo Coordinador: Sólo puede existir un nodo coordinador en una red ZigBee, su misión es la de inicializar la red, seleccionar el canal apropiado de operación, permitir a otros dispositivos unirse a la red y gestionar las rutas. En nuestro caso, tal y como se ha dicho, se utilizará este nodo como elemento colector. Dada su importancia en la red este dispositivo no puede utilizar modos de reposo pues siempre debe estar realizando tareas de supervisión y control.
- Nodo *Router*: Puede haber varios *routers* en la red, su misión es la de reenviar los datos desde los dispositivos finales, almacenan también las rutas y al igual que el coordinador no pueden entrar en modos de reposo. Dado el tamaño de nuestra red, en principio, no se utilizará ningún nodo como nodo repetidor, aunque su configuración es posible y con el software desarrollado se podría hacer de forma dinámica mediante la modificación del parámetro SM que se verá más adelante.
- Dispositivos finales: Son los nodos sensores cuya misión es generar o recibir información para ser distribuida por la red. En nuestro proyecto contamos con 9 nodos finales cada uno de ellos dotados de 4 sensores de temperatura.

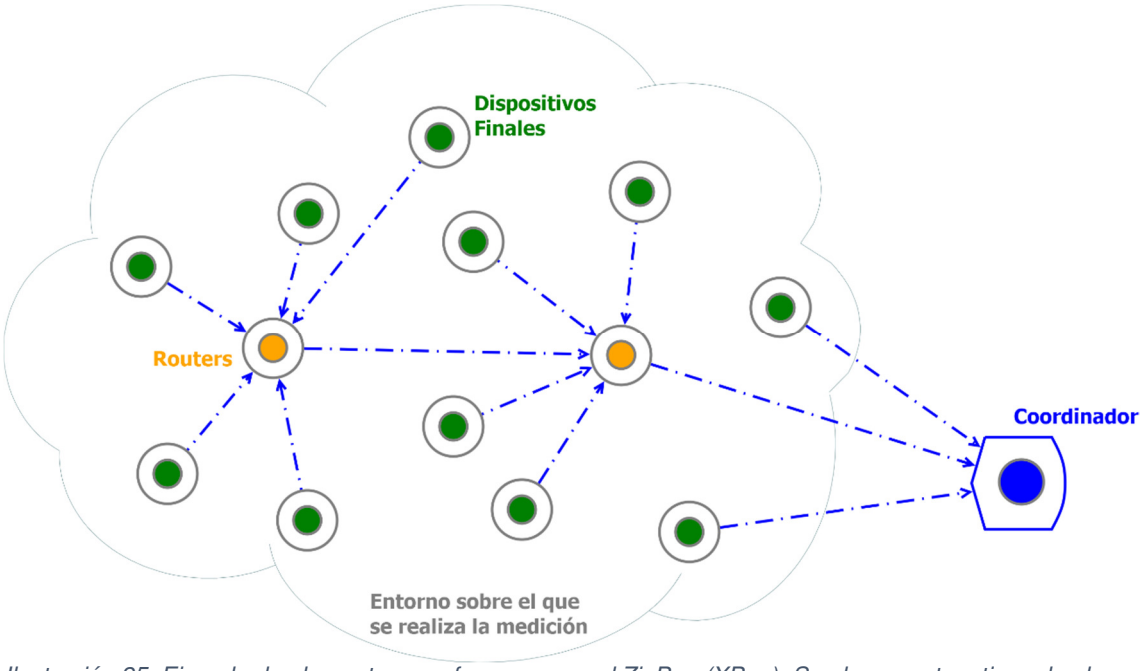


Ilustración 25. Ejemplo de elementos que forman una red ZigBee (XBee). Se observan tres tipos de elementos el nodo coordinador (Azul), los routers (naranja) y los dispositivos finales (verde)

d) SH y SL. Serial Number High y Low (Número de serie del nodo)

Aunque se trata de un registro simplemente de lectura, se quiere indicar aquí que es utilizado en este proyecto para posibilitar la configuración remota de los nodos, de forma que si desde el software desarrollado, como se verá más adelante, se quiere configurar un nodo debemos indicar el número de serie de dicho nodo.

e) DH y DL. Destination Number Hig y Low (Dirección del nodo destino)

Se trata de dos registros que indican la dirección completa del nodo destino con el cual el nodo a configurar se puede configurar, el rango de valores que admiten dichos registros es:

0 - 0xFFFFFFFF.

Así por ejemplo si configuramos con un valor:

0x000000000000FFFF.

Indicamos que queremos una comunicación broadcast con toda la red.

Si por el contrario configuramos los registros con un valor:

0x0000000000000000.

Indica que queremos comunicarnos con el nodo coordinador.

f) NI. Node Identifier (Identificador del nodo)

Se trata de una cadena que admite de 0-20 caracteres ASCII, para este proyecto se nombrara como COORDINADOR al nodo colector el resto de nodos se nombrarán como REMOTO_1, REMOTO_2, hasta REMOTO_9.
Este identificador se usará en el software de almacenamiento y visualización de los datos generados.

g) AP. API Enable (Habilita el modo de transmisión basado en paquetes)

Los valores que admite el registro son:

0 – Modo transparente.
1 – Modo API habilitado.
2 – Modo API habilitado con carácter de escape.

Los datos desde los nodos pueden ser enviados o bien en modo transparente, o bien empaquetados dentro de una trama.
En el modo transparente el dato se envía sin ningún añadido, la idea es que se envía el comando o los datos como si se estuviesen mandando directamente por el puerto serie. Este modo de comunicación, aunque es muy sencillo, tiene el inconveniente de que en los datos enviados no viene información sobre el emisor, por tanto, no se puede utilizar con redes de varios nodos ya que no existe la forma de conocer su origen.
El modo API por su parte, tanto los comandos como los datos se envían dentro de un paquete con una estructura y protocolos definidos, esto permite en el receptor conocer quién le ha enviado la información.
La diferencia entre el modo API con y sin escape es que en el último se incluye un carácter de escape al final de la trama (0x7D), esto hace que si existe un problema en los bytes dentro de un paquete se conozca perfectamente donde empieza el siguiente y no se tenga que desestimar el paquete siguiente lo que ocurre en el modo sin carácter de escape.
Los nodos en este proyecto se configurarán en modo AP=2, no obstante, el modo AP=1 también es posible sin realizar ninguna modificación en el software salvo la modificación del registro de configuración del propio nodo.

h) SM. Sleep Mode (Modos de reposo)

Este registro admite los siguientes valores:

0 – No Sleep. Configurado como Router.
1 – Pin Sleep. Un valor alto lo pone en reposo y bajo lo activa.
4 – Sleep cíclico.
5 – Sleep cíclico con pin de despertar.

En nuestro proyecto se ha utilizado el modo de *sleep* cíclico en el que los nodos cíclicamente alternan entre modos de actividad y de reposo, dada la importancia de este punto y el hecho de que afecte a otros parámetros hace que se comenten con mayor detalle los modos de reposo en el punto siguiente.

i) SP. Sleep Period (Periodo de reposo)

Antes de indicar los valores posibles para los registros que afectan al comportamiento de los modos de reposo, conviene explicar brevemente cómo funciona este modo de trabajo.

Los modos de reposo se utilizan para reducir el consumo de energía y por tanto incrementar la vida de las baterías. Se basan en la utilización de lo que se conoce como mensajes indirectos, este tipo de mensajes se producen cuando, durante uno de esos ciclos de "sueño" el coordinador o *router* debe almacenar el mensaje para ser enviado al nodo cuando este despierte, esto es posible porque cuando se produce este despertar de un nodo, éste, manda un mensaje a su dispositivo padre para conocer si hay algún dato o petición pendiente.

El nodo coordinador sólo es capaz de guardar un mensaje indirecto para cada nodo, es decir, no puede guardar una cola de mensajes por nodo, además el coordinador almacena el mensaje durante un periodo de 2'5 veces el tiempo marcado en este parámetro SP (*Time Before Sleep).*
Un dispositivo remoto se activa una vez por periodo y envía una solicitud al coordinador para conocer si existe un mensaje indirecto para él. Si no existe, el nodo volverá al modo sueño durante otro nuevo ciclo, este proceso consume aproximadamente 30 ms por ciclo. Si por el contario hay un mensaje pendiente para el nodo remoto, el coordinador transmitirá este mensaje directo, y el nodo remoto permanecerá despierto de acuerdo con el parámetro ST (*Time Before Sleep)*, tiempo en el cual debe ser capa de recibir todo el mensaje.

Así pues, centrándonos en el valor de SP tenemos que los valores con los que se puede configurar este registro son:

0x20-0xAF (el valor decimal se multiplica x10 para obtener el valor en mili segundos, por tanto, desde 320ms a 28000ms).

Y tal y como hemos visto, define el periodo de tiempo que un dispositivo final está en modo reposo.

j) SN. Number of Cyclic Sleep Periods (Número de ciclos de reposo)

El registro admite el siguiente rango de valores:

0x1 – 0xFFFF (1 – 65535).

Indica el número de ciclos (SP) que el dispositivo va a considerar para los periodos de reposo y por ende para los *timeouts*, esto se realiza mediante la configuración del parámetro SO que se verá más adelante y permite la extensión de los periodos de reposo de los nodos.
En nuestro proyecto se ha tomado el valor por defecto, 1, aunque se puede ajustar libremente.

k) ST. Time Before Sleep (Tiempo antes de dormir)

Los valores que admite este registro oscilan entre:

0x1 – FFFE (1- 65534 ms).

Define el periodo de para la recepción del mensaje indirecto antes de entrar en el modo sueño tal y como se ha explicado en el apartado dedicado al registro SP.

l) *SO. Sleep Options (Opciones de sueño)*

Los valores que admite son:

2 – Siempre despierto para todo ST.
4 – Habilita el modo extendido de sueño (duerme durante SP*SN milisegundos).

Tal y como se ha dicho nosotros podemos usar libremente cualquiera de los dos modos según convenga.
Un detalle crucial que hay que destacar aquí, es que el diseño realizado va a marcar absolutamente la relación entre la cantidad de tiempo que un nodo permanezca dormido y la sincronización en el dato enviado.
Lógicamente si un nodo está dormido y el nodo coordinador le hace una petición, este no va a responder hasta que despierte y pregunte si tiene algún envío pendiente. Si este tiempo de sueño es muy elevado la sincronización de los datos dependerá del momento en que se hace la petición puesto que no es lo mismo hacerla un poco antes de que el nodo despierte del modo sueño a hacerla justo cuando acaba de entrar en él.
Una opción que admiten este tipo de módulos es la activación por medio de una entrada digital, sin embargo, esto se descartó debido a la necesidad de un microcontrolador y electrónica adicional para la sincronización de este modo de sueño. Por tanto, contra más sincronismo queramos en las medidas, menor deberá ser el modo sueño, por lo que habrá que llegar a un compromiso entre sincronismo frente a duración de las baterías.

m) SO. Parámetros de diagnóstico.

Los módulos XBee también incorporan registros de lectura que proporcionan información de diagnóstico, en este proyecto se han utilizado dos de ellos:

HV. Hardware Version. Se ha utilizado para determinar el modelo de nodo utilizado ya que, según la versión, se incorporan uno u otros parámetros. Así si se intentase configurar un nodo con un parámetro que no posee, el sistema daría error. Mediante la lectura de la versión de hardware se previene de que este error pueda ocurrir.

%V. Supply Voltage. Indica la tensión con la que está siendo alimentado el nodo. Dada que es la misma tensión de alimentación que del resto de componentes es importante conocerla debido al sensor de temperatura elegido, no obstante, esto se comentará con mayor detalle en siguientes secciones.

5.1.2 Sensores de temperatura.

Los sensores de temperatura se presentan como otro elemento clave a la hora de diseñar el sistema ya que su elección marca la precisión de las medidas obtenidas. A continuación, se muestra un breve resumen de los sensores que han sido analizados y las conclusiones a las que se ha llegado.

5.1.2.1 Requerimientos previos

En primer lugar, se fija como objetivo muy deseable utilizar sensores analógicos de bajo voltaje, preferiblemente un máximo 3.3v para aprovechar la alimentación del nodo, e idealmente con salida de hasta 1.2v para ser compatibles con la tensión de operación de las entradas de nuestros módulos XBee, de otra forma sería necesaria la inclusión de otro microcontrolador, lo cual, entre otras cosas, afectaría a los consumos y a la gestión de los modos de reposo. El rango de temperaturas a medir siempre estará dentro del rango de -20ºC a 40ºC (Los valores históricos que se han registrado en el observatorio están dentro de este rango con un margen de seguridad de varios grados centígrados).

Una vez establecida la necesidad de utilizar sensores de temperatura analógicos se estima la mínima precisión necesaria, nos basamos en los estudios presentados anteriormente para afirmar que se espera que la variación del *seeing* por turbulencia sea del orden de 0,10"/ºC· ΔT a 0,15"/ºC· ΔT, siendo ΔT la diferencia de temperatura entre dos puntos medidos, sobre este punto hay que destacar que por fabricación la distancia entre un par de sensores de cada nodo no puede ser mayor de 6 cm, lo cual es muy útil para medir diferencias entre una superficie y su capa de aire inminentemente superior, sin embargo, si se quiere que la distancia entre sensores sea mayor hay que utilizar dos nodos y separarlos la distancia deseada, así pues la versatilidad de este sistema, nos permitirá realizar estudios como el anteriormente comentado para el Observatorio Nacional de San Pedro Mártir [30].

Los datos de *seeing* con los que trabajamos en el observatorio son del orden de la décima de segundo de arco y nunca superiores a centésima de segundo de arco, esto hace que la utilización de sensores de baja precisión origine que los valores pierdan su validez, así por ejemplo si un sensor tiene una precisión en la medida de 2ºC el error máximo que se puede introducir de diferencia entre dos sensores sería de 4ºC, lo que se traduce en un error máximo estimado de 0.6", que invalidaría totalmente el estudio.

Siguiendo este razonamiento tenemos las siguientes conclusiones:

- Si queremos una precisión del orden de décima en las medidas de seeing se obliga a que la precisión en la medida del sensor de temperatura nunca sea inferior a 0.3ºC (aprox.), ya que esto origina un error máximo entre medidas de dos nodos de 0,6ºC y por tanto un error máximo de *seeing* de 0,15"/ºC·0'6ºC= 0'09".

- Si por el contrario queremos una precisión en el orden de la centésima de segundo de arco los valores en precisión de los sensores que se requieren son

del orden de los 0,03ºC, lo que originaría un error máximo en la determinación del *seeing* del orden de 0,15"/ºC·0'06ºC= 0'009".

Finalmente, dos aspectos importantes a destacar son, primero que los conversores AD de los módulos XBee dotados de una resolución de 10bits, esto es 1024 posibles valores, si el rango de medidas que admite el módulo XBee es de 1.2 voltios implica que la máxima resolución que se puede obtener, debido al paso de cuantificación, es de 1.2v/1024≈1,17 mv.
En este rango de conversiones hay que prestar especial atención a las tensiones y rangos de medida de los sensores. Y, en segundo lugar, que cada nodo cuenta con 4 entradas analógicas, por tanto, para aprovechar al máximo el potencial que ofrece cada nodo se decide equiparlos con 4 termistores a cada uno.

5.1.2.2 Sensor TPM36

Se trata de un sensor de temperatura analógico de bajo voltaje, tensión de alimentación entre 2,7v y 5,5v con una precisión de ± 2ºC, una linealidad típica de 0,5ºC, una variación de 10mv/ºC y un rango de medidas que oscila desde los -40ºC a 125ºC [73].
Dentro del rango de temperaturas que nos hemos fijado la tensión de salida del sensor está dentro del margen de 1.2v que admiten las entradas de los módulos
(y con bastante margen).

Por parte del conversor AD esta variación de la tensión frente a la temperatura hace que la máxima resolución sea de:

$$Res = 1{,}17mv/10mv/ºC=0'117ºC$$

Los sensores se han probado con buenos resultados, no obstante, su valor de precisión de 2ºC, tal y como se ha comentado en los requerimientos previos, invalidan totalmente las medidas obtenidas por lo que se desestima la utilización de este sensor.

5.1.2.3 Sensor LMT84

Se trata de otro sensor de temperatura analógico de bajo voltaje, su precisión típica es de 0.4ºC con una variación de -5,5mv/ºC y su rango de medición va desde los -40ºC hasta los 150ºC, para esos valores sus salidas de tensión son de 1,3v a 0v respectivamente [74].
Se trata de un sensor perfectamente válido para este proyecto puesto que, aunque esta precisión en temperatura se traduzca en valores de una décima de segundo de arco, podríamos considerarlo dentro de los límites establecidos tal y como se ha explicado.
En cuanto al conversor AD como hemos visto cuantifica en saltos de 1,17mv lo que implica que la resolución obtenida sería de:

$$Res = 1{,}17mv/5{,}5mv/ºC=0'3ºC$$

También dentro de los márgenes fijados (0'3ºC, implica un error máximo entre dos medidas de 0'6ºC y tal y como se ha explicado generaría un error de 0,09").

5.1.2.4 Termistor

Con el fin de que el sensor no sea el elemento limitante en la precisión de las medidas, se estudia la utilización de un termistor NTC de alta precisión, se trata de un termistor de la serie PR103J2 de la casa US Sensor/Littelfuse Inc [75].
La característica principal de este sensor es que su tolerancia en la resistencia es de 0,05ºC con lo que la precisión en los valores de seeing sería de centésima de segundo de arco.
El rango de medidas de este termistor abarca desde los -55ºC hasta los 80ºC lo que encaja perfectamente en los rangos que nos hemos marcado.

Por otra parte, el fabricante proporciona una hoja Excel dónde se muestra una tabla con la variación de la resistencia en función de la temperatura, así para los valores que nos hemos fijado tenemos:

$R_{-20ºC}$ = 97081,38 Ω
$R_{40ºC}$ = 5326,04 Ω

Siguiendo con este razonamiento sabemos que la tensión de alimentación es como máximo de 3,3v y queremos conseguir también un valor máximo de 1.2v en las entradas del módulo, esto lo podemos conseguir mediante un simple divisor de tensión. Por ello, aunque se mostrará en el esquema eléctrico, se ha serializado con el termistor una resistencia de 210kΩ y muy baja tolerancia ±0,1% de esta forma las entradas en voltaje obtenidas para nuestros márgenes son:

$T_{-20ºC}$ = 3,3v * 97081,38 Ω / (97081,38 Ω+210000 Ω) = 1’04v
$T_{40ºC}$ = 3,3v * 5326,04 Ω / (5326,04 Ω+210000 Ω) = 0’081v

Esto significa que hay una variación de 0,96v en estos 60ºC por tanto una variación de 16 mv/ºC, esto hace que la resolución del conversor AD nos dé un valor de:

Res = 1,17mv/16mv/ºC=0’07ºC

Finalmente destacar que la resistencia en serie al termistor, tal y como se ha dicho, se ha seleccionado baja tolerancia ±0,1% lo que supone un error del mismo orden en las medidas de voltaje y por tanto despreciable.

5.1.2.5 Elección del sensor

Tal y como se ha visto en el apartado anterior tanto el sensor LMT84 como el termistor de baja tolerancia, encajarían en los objetivos del proyecto, sin embargo, hay tres factores a tener en cuenta.

- El primero es que la precisión en los resultados es de un orden mayor con el termistor.
- El segundo es que el precio del termistor (junto con la resistencia de alta precisión) es casi un factor x10 más elevado que el del LMT84, este valor hay que escalarlo al hecho de que necesitamos entre 36-40 sensores.
- El tercero es el consumo, aunque no se ha hablado en el punto anterior, el fabricante del LMT84 habla de un consumo de 50µA cuando el sistema está midiendo y 5.4µA cuando está inactivo. Por su parte, una estimación grosera

del consumo debido a los elementos resistivos con el termistor nos habla de un consumo continuo, en el peor de los casos de: I = 3,3v / (5326,04 Ω + 210000 Ω) = 15,3 µA.

Así pues, se concluye que todas las desventajas del termistor son asumibles frente a la gran ventaja en precisión que proporciona, por tanto, se decide utilizar dicho sensor como elemento de adquisición de datos para el presente proyecto.

5.1.2.6 Cálculos y calibración de los sensores

Sabemos que un termistor cambia su resistencia en función de la temperatura, uno de los problemas que origina la utilización de termistores es la falta de linealidad en su comportamiento, por lo que el cálculo del valor de temperatura en función de su resistencia se complica.
Una forma de solventar este problema sería linealizarlo mediante el circuito correspondiente, pero, lógicamente, se pierde la tan buscada precisión en las medidas.
Otra forma sería, dado que el fabricante nos proporciona el parámetro B0/50, la utilización de la ecuación de Steinhart-Hart [76] para convertir los valores de resistencia en temperatura, sin embargo, se ha podido comprobar gracias a la tabla Resistencia/Temperatura que proporciona el propio fabricante, que los resultados no eran del todo satisfactorios, pues se observaban errores de conversión resistencia-temperatura al comparar los valores calculados con los proporcionados por dicha tabla, esto se atribuye a que, debido a la falta de los parámetros de segundo orden, se ha tenido que utilizar una versión simplificada de la ecuación.
Finalmente, la solución encontrada ha sido la de aprovechar los valores que se proporcionan en la tabla para realizar una regresión cuadrática y así encontrar la curva que mejor se ajuste a la facilitada por el fabricante, para ello se ha hecho uso de Matlab, concretamente de la herramienta *curve fitting,* obteniéndose una ecuación sencilla que se ajusta a los datos aportados y cuyo error cuadrático medio (RMSE) es del orden de la centésima de ºC

$$Temp^{\circ}C = 574'1 \cdot Resistencia^{-0'1242} - 158$$

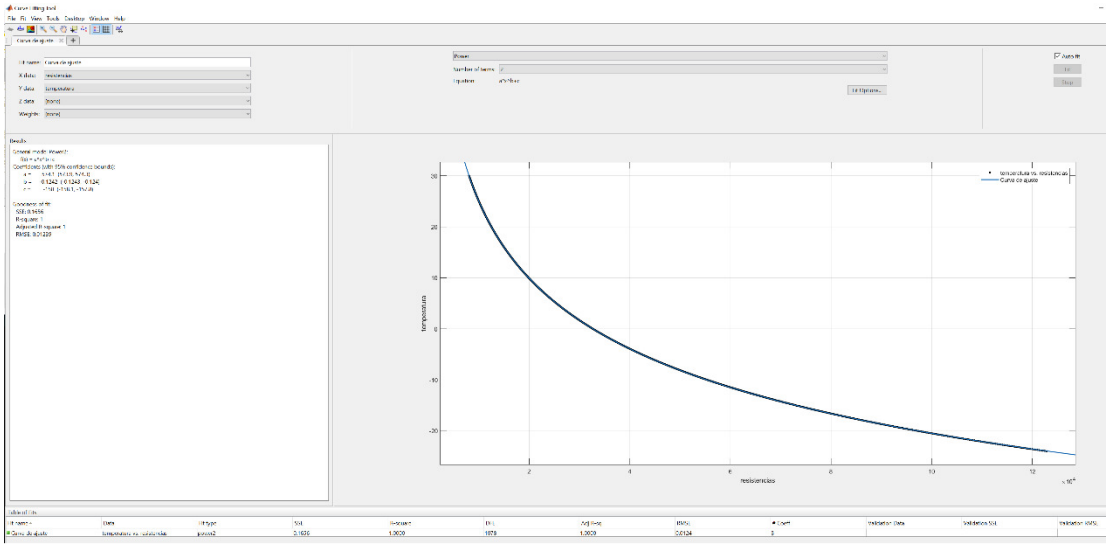


Ilustración 26. Curva de ajuste a los valores de resistencia de los termistores proporcionados por el fabricante mediante el uso de Matlab y regresión cuadrática

Otro aspecto a tener en cuenta cuando se utilizan termistores es que, al tratarse de un divisor de tensión, el valor en voltaje a la entrada depende de la tensión de suministro y que esta puede variar por agotamiento de la batería. Por ello aprovechando que dicha tensión es la misma que la de alimentación al módulo, antes de realizar la conversión de la entrada a resistencia, el módulo realiza la medición de dicha tensión de suministro que será la que se utilice por el software para los cálculos, es decir, conociendo la tensión de entrada y la de alimentación aplicando la ley de Ohm es muy sencillo conocer la resistencia del termistor.

Finalmente indicar que se han medido los valores de tensión de entrada a lo largo de diferentes pruebas y no se han encontrado diferencias significativas con los resultados obtenidos por el software, dichas diferencias eran del orden de 1mv, por tanto de la misma magnitud que el error de cuantificación del conversor AD, por ello se considera que dichas mediciones son coherentes con los datos aportados por el fabricante y que cumplen los requerimientos fijados para este proyecto, no obstante, sí que se recomienda en plazos de realización mayores, y con mejores sistemas de calibración un control más preciso de la precisión en las medidas.

5.1.3 Fabricación de la PCB.

Una vez que los dos componentes fundamentales fueron definidos, se hizo un diseño conceptual de todos los componentes electrónicos que conformarían el nodo para su correcto funcionamiento (más adelante se hablará del software elegido).
Evidentemente se hace necesaria una fuente de suministro, aunque más adelante se tratará con mayor profundidad, se decidió utilizar una pila de 9v y dado que los módulos trabajan a una tensión de suministro máxima de 3’3v se hace necesaria la utilización de un regulador de tensión y un condensador para el filtrado de dicho suministro.
Desde el punto de vista hardware el nodo en sí mismo es muy sencillo, algo que es coherente con el propio concepto de las WSN, los elementos que se han incorporado son (En el resumen económico se incorporarán más detalles para su posible replicación):

- Sistema de conexión del sistema de alimentación:
  - Bloque terminal de 2 polos, 2,54mm para PCB.
  - Conector para pila de 9v y 4”.
  - Micro interruptor de palanca.
  - Pila alcalina 9v.
- Sistema acondicionador de tensión:
  - Condensador electrolítico 16V 100uF.
  - Regulador de tensión LDO 3.3v 1A.
- Nodo y sistema de acondicionamiento para PCB
  - Nodo Digi XBee S2C.
  - Socket de 10 pines.
- Elementos de medida:
  - Termistores PR103J2 10K 0,05ºC.
  - Resistencia baja tolerancia 210KΩ.

Cabe aquí decir que, aunque la idea inicial era fabricar nosotros mismos nuestros propios circuitos, incluso se llegó a fabricar una insoladora casera basada en tiras leds, durante la investigación para la realización de las mismas se encontró una página web de la empresa JLCPCB [77] a la que se podían enviar nuestros diseños (mediante los archivos *gerber* generados en el proceso de diseño), de esta manera, dicha empresa fabrica las PCBs a unos precios (se darán detalles en la valoración económica) y acabados difícilmente alcanzables con los medios limitados con los que contamos, por tanto, se toma la decisión de fabricar las placas a través de esta web. Aunque se incorporan los esquemas del diseño en los anexos, se muestra aquí el concepto de diseño y su implementación final.

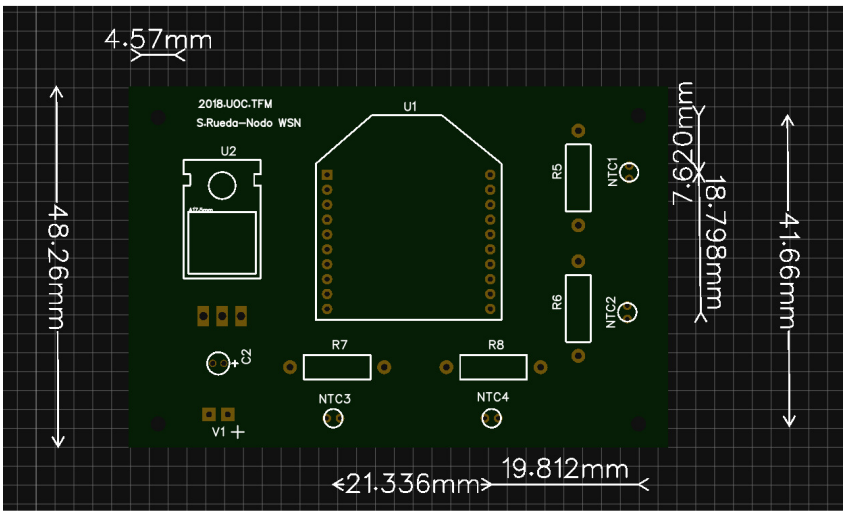


Ilustración 27. Vista superior de la PCB diseñada para un nodo sensor mediante el software EasyEDA

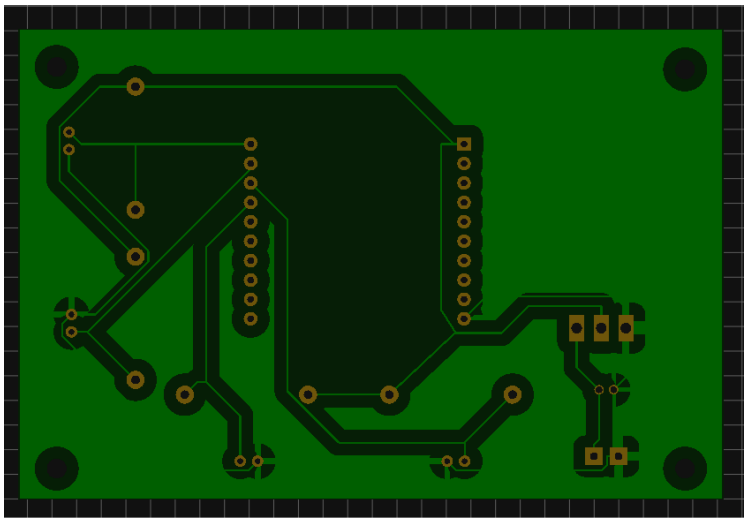


Ilustración 28 Vista inferior de la PCB diseñada para un nodo sensor mediante el software EasyEDA

5.1.4 Gateway. Nodo colector

Tal y como se ha comentado, se deseaba utilizar una de las tarjetas de desarrollo que se incorporan en el *Digi Mesh Kit* para, vía comunicación serie, servir de interface entre la red RF y la máquina donde se ejecutará el software de adquisición diseñado. También es importante indicar que se pretendía que el equipo dónde se ejecuta dicho código fuera un elemento flexible y que por tanto admitiese distintas posibilidades, es decir, que no estuviese supeditado a una única plataforma hardware o software, por lo que se ha tratado de hacer un diseño lo más abierto posible y así el sistema puede funcionar usando tanto un PC de sobremesa con distintos sistemas operativos, como una *raspberry pi* (modelos 2, 2b y 3).
Finalmente indicar que en la implantación definitiva se utilizará una *raspberry pi model 2b*. Los motivos para la elección de este hardware se explicarán a continuación.

5.1.4.1 PC de sobremesa

Se ha probado con éxito todo el sistema de adquisición trabajando con un nodo colector conectado a un PC (se ha probado en varios equipos) y con los sistemas operativos Windows 10, Ubuntu (versión 14LTS y 16LTS).
El software desarrollado es, por tanto, compatible con los sistemas operativos mencionados, no obstante, hay que aclarar que se requiere la instalación de ciertos programas y librerías, y la modificación de parámetros de configuración dicha instalación es dependiente del sistema operativo.
En los anexos se incluye la configuración para equipos basados en Windows y basados en Unix/Linux (desde las nuevas versiones de macOS también debería ser posible la utilización del software)

5.1.4.2 Raspberry pi 2 model b

La implementación final del sistema colector en el observatorio se basará en una r*aspberry pi 2 model b,* se decide utilizar este hardware más sencillo en lugar de un PC, por diversos motivos, pero principalmente por la portabilidad que ofrece, ya que es muy cómodo transportarlo, instalarlo y dejarlo de forma permanente en el

observatorio, además también esto supone una utilización más eficiente de los recursos ya que con el uso de un PC para esta tarea de forma dedicada, aunque es posible, se estarían desaprovechando recursos, finalmente indicar que al ser un elemento que va a estar cerca de la instrumentación se precisa que genere el menor calor posible y por supuesto no emita radiación luminosa que pueda ser captada por los sistemas.
Finalmente indicar que la elección del modelo 2b en lugar del 3 se basa en una cuestión meramente práctica, ya que este hardware está disponible en el observatorio y no se hace necesaria su adquisición.

5.1.5 Sistema de alimentación. Opciones

Tal y como se ha indicado, en las redes de sensores la elección de la batería suele ser un hecho vital, ya que su duración, en caso de que la energía no provenga de una fuente inagotable, marcará la vida útil del sensor o al menos el tiempo de funcionamiento de manera autónoma.
La capacidad total de una batería se mide en miliamperios-hora (mAh) o en amperios-hora (Ah), así por ejemplo una pila de 4Ah es capaz de suministrar 4 A durante una hora ó 1 A durante 4 horas.

A la hora de diseñar el sistema de baterías debemos tener en cuenta, al menos, dos aspectos principales [78]:

- Las condiciones de temperatura pueden afectar al rendimiento de las baterías, así a modo general, se puede decir que en lugares donde las temperaturas no son demasiado elevadas o demasiado frías, las baterías se conservan bien, por el contrario, en entornos muy calientes o muy fríos pueden perder efectividad y no ser capaces proporcionar el rendimiento esperado. El observatorio es un centro donde las temperaturas suelen ser muy suaves en verano con lo que este punto no afectaría en el presente TFM, pero sí que son extremadamente frías en invierno por tanto es un punto a tener muy en cuenta, además, si en algún momento se decide realizar la medición en algún entorno más caliente (sala de generadores, cuadros eléctricos, etc.) este rendimiento, tal y como se comenta también podría verse afectado.

- La capacidad real de las baterías no es lineal y depende de la corriente a suministrar, así pues, a pesar de que el fabricante nos proporciona la carga nominal conviene tener en cuenta que la real dependerá de la corriente a suministrar. Aunque se verá en el capítulo dedicado a la visualización de datos, indicar aquí que se puede monitorizar en tiempo real la batería de cada nodo para prever cuando será necesario su recambio.

Finalmente indicar que los módulos XBee necesitan una alimentación de 2'8 v a 3'4 v y su consumo es de [72]:

- 45 mA máximo en fase de transmisión
- 50 mA máximo en fase de recepción
- <10 µA cuando el sistema está en *sleep mode*

5.1.5.1 Baterías alcalinas

Es el tipo de batería más común para uso doméstico, sus principales ventajas son, su bajo coste, el hecho de que se puedan adquirir fácilmente y que son ideales para aplicaciones de baja demanda de corriente a temperatura ambiente.
No obstante, presentan dos inconvenientes, el primero es que su rendimiento decae mucho para alta corriente y segundo y más importante en nuestro caso, que su rendimiento empeora mucho a temperaturas muy frías, algo que como se ha dicho, ocurre todos los inviernos en el observatorio.

5.1.5.2 Baterías de litio

Existe una amplia variedad de baterías de litio en el mercado, sin embargo, todas ellas presentan un comportamiento común, y es que funcionan mejor a bajas temperaturas que las baterías alcalinas, sin embargo, su precio se multiplica en un factor 3 ó 4 si lo comparamos con dichas pilas alcalinas.

5.1.5.3 Baterías de Níquel-Metal-Hidruro

Se trata de baterías recargables de gran calidad, una de sus ventajas es que no pierden prestaciones tras muchas recargas, algo que si ocurre con la batería de Níquel-Cadmio, se pueden considerar como una mejora de las alcalinas ya que se comportan mejor a bajas temperaturas y con alta demanda de corriente. En cuanto a su precio se pueden encontrar, aproximadamente, al doble de precio que las baterías alcalinas.

5.1.5.4 Batería elegida

Si se hace un balance calidad-precio se consideran óptimas las baterías de Níquel-Metal-Hidruro y se recomienda que sean estas las incorporadas para una versión definitiva del sistema de sensado, sin embargo, dado que actualmente las condiciones climáticas no son severas y al estar en una fase inicial en la que se prevé no hacer un uso todavía totalmente desasistido, se ha decidido adquirir un pack de pilas alcalinas de 9v y 640mAh para el conjunto de sensores.

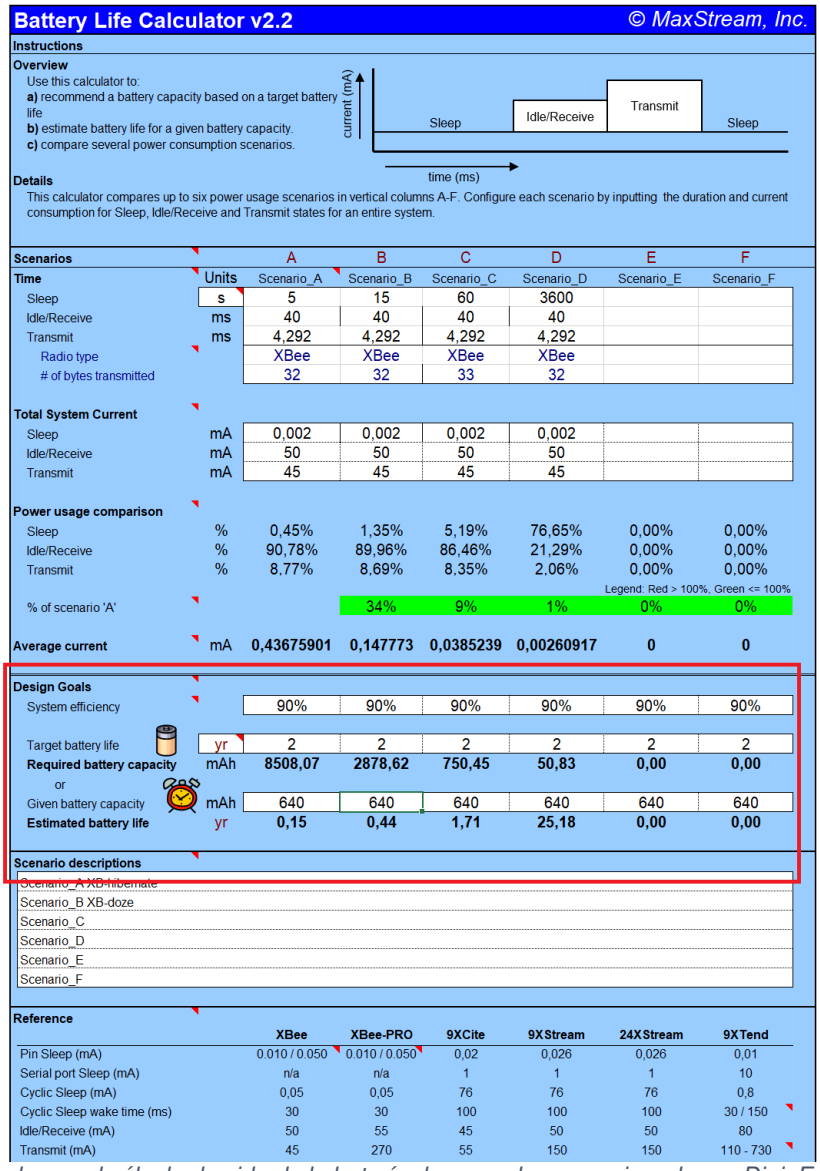
Se ha explicado con anterioridad que la tensión de suministro del tipo de batería elegida está por encima de la de trabajo de un nodo XBee, por tanto, dichas baterías son conectadas a un sistema de filtrado y regulación de voltaje a 3,3v para un correcto funcionamiento de todo el circuito.

Finalmente añadir que la casa Digi proporciona una hoja Excel para hacer una estimación de la duración de las baterías (Los datos de consumo, tiempos de transmisión, etc. También son proporcionados por Digi en la propia hoja Excel [79]).

Como se observa en la imagen siguiente, se ha hecho el cálculo para diferentes valores de configuración del tiempo en el que el nodo se mantiene en modo sueño, yendo desde los cinco segundos hasta la hora.
Vemos que en la configuración más exigente 5 segundos la vida de la batería instalada se calcula de aproximadamente 0’15 años (55 días aproximadamente), si el nodo

entrase en reposo durante 15 segundos la duración de la batería asciende a 0’44 años (160 días aproximadamente), mientras que si la duración de este modo de baja energía es de un minuto la batería casi duraría 2 años, finalmente, en periodos de tiempo mayores el consumo pasaría a ser un problema de segundo o tercer orden ya que deja de ser un aspecto crítico.

Finalmente indicar que estos valores proporcionados se deben tomar con precaución y cautela, y se debe tener en cuenta además el consumo de los termistores, resistencias, etc. (Tal y como se comentó anteriormente, el consumo del conjunto sensor resistencia-termistor no debe ser superior a 15uA). Es por ello, tal y como se ha dicho, que el sistema de visualización incluye una medición periódica de la vida del estado de la batería (se muestra el voltaje suministrado al nodo).



Battery Life Calculator v2.2 — © MaxStream, Inc.

Instructions

Overview

Use this calculator to:
a) recommend a battery capacity based on a target battery life
b) estimate battery life for a given battery capacity.
c) compare several power consumption scenarios.

Details

This calculator compares up to six power usage scenarios in vertical columns A-F. Configure each scenario by inputting the duration and current consumption for Sleep, Idle/Receive and Transmit states for an entire system.

Scenarios		A	B	C	D	E	F
Time	Units	Scenario_A	Scenario_B	Scenario_C	Scenario_D	Scenario_E	Scenario_F
Sleep	s	5	15	60	3600		
Idle/Receive	ms	40	40	40	40		
Transmit	ms	4,292	4,292	4,292	4,292		
Radio type		XBee	XBee	XBee	XBee		
# of bytes transmitted		32	32	33	32		
Total System Current							
Sleep	mA	0,002	0,002	0,002	0,002		
Idle/Receive	mA	50	50	50	50		
Transmit	mA	45	45	45	45		
Power usage comparison							
Sleep	%	0,45%	1,35%	5,19%	76,65%	0,00%	0,00%
Idle/Receive	%	90,78%	89,96%	86,46%	21,29%	0,00%	0,00%
Transmit	%	8,77%	8,69%	8,35%	2,06%	0,00%	0,00%
						Legend: Red > 100%, Green <= 100%	
% of scenario 'A'			34%	9%	1%	0%	0%
Average current	mA	**0,43675901**	**0,147773**	**0,0385239**	**0,00260917**	**0**	**0**
Design Goals							
System efficiency		90%	90%	90%	90%	90%	90%
Target battery life	yr	2	2	2	2	2	2
Required battery capacity	mAh	**8508,07**	**2878,62**	**750,45**	**50,83**	**0,00**	**0,00**
or							
Given battery capacity	mAh	640	640	640	640	640	640
Estimated battery life	yr	**0,15**	**0,44**	**1,71**	**25,18**	**0,00**	**0,00**

Scenario descriptions

Scenario_A XB-hibernate
Scenario_B XB-doze
Scenario_C
Scenario_D
Scenario_E
Scenario_F

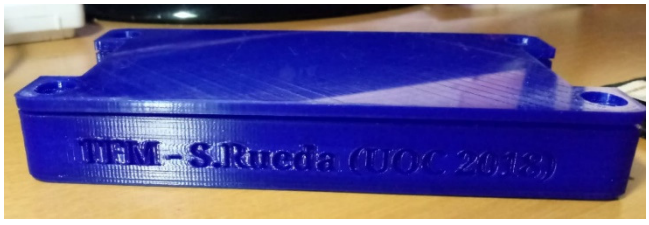
Reference

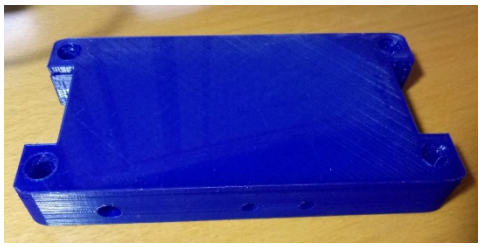
	XBee	XBee-PRO	9XCite	9XStream	24XStream	9XTend
Pin Sleep (mA)	0.010 / 0.050	0.010 / 0.050	0,02	0,026	0,026	0,01
Serial port Sleep (mA)	n/a	n/a	1	1	1	10
Cyclic Sleep (mA)	0,05	0,05	76	76	76	0,8
Cyclic Sleep wake time (ms)	30	30	100	100	100	30 / 150
Idle/Receive (mA)	50	55	45	50	50	80
Transmit (mA)	45	270	55	150	150	110 - 730

Ilustración 29. Excel para el cálculo de vida de la batería de un nodo proporcionado por Digi. En el recuadro rojo se observa la capacidad de la batería para un objetivo de 2 años y justo debajo la duración de la batería estimada según las adquiridas en este proyecto (640 mAh). Las columnas marcan distintos escenarios con distinta duración del periodo de sueño del nodo

5.1.6 Diseño caja estanca para albergar la electrónica

Tal y como se ha dicho, uno de los objetivos del proyecto era atender también a esa vertiente con un carácter más industrial, esto implica el diseño de un producto que pueda ser replicado y por tanto fabricado para poder ser utilizado para tareas de monitorización de temperaturas en entornos que no tienen por qué tener una gran similitud con el observatorio.
Para completar esta vertiente industrial, por tanto, no solo se fabricaron desde cero las placas de circuito impreso, también se han diseñado y fabricado las cajas que contendrán dicha electrónica y que serán instaladas en el observatorio.
Aunque sobre el proceso de diseño se hablará más adelante en el parte software, sí que, a continuación, se dará una breve pincelada de cómo se han fabricado las cajas y se mostrará el resultado obtenido.
Aprovechando que el observatorio cuenta con una impresora 3D, se han creado 9 cajas que contendrán la electrónica de los nodos sensores.
Dichas cajas han sido fabricadas en PLA [80], se trata de un material altamente versátil fabricado al 100% a partir de productos renovables, en cuanto a su tamaño, presenta unas dimensiones de 155x80x20mm y están formadas por dos componentes principales, la parte que contiene la electrónica y la tapa. Como se verá en las imágenes siguientes, tras varias iteraciones se ha llegado a un diseño que facilita que los sensores puedan ser sacados al exterior, bien para ser dejados al aire o para ponerlos en contacto con alguna superficie, de esta forma se facilita la consecución de los requerimientos planteados, como es el hecho de ser capaces de medir la temperatura de una superficie y la capa de aire más próxima a la misma.
En el diseño de la caja también se contempla un compartimento para la pila de 9v que evita que esta se pueda mover una vez que la caja esté cerrada, y la instalación de un micro interruptor para dar tensión al conjunto.
Indicar que la caja se ha tratado de diseñar lo más estanca posible, por tanto, se incorporado un pequeño zócalo de acoplamiento entre la caja y la tapa, y se posibilita su cierre con tornillos M8 y rosca hexagonal.
Otro de los retos que ha habido que superar es cómo instalar la caja en la cúpula, sobre todo sabiendo que si una caja se desprende el resultado puede ser catastrófico, pues podría impactar contra elementos extremadamente caros y frágiles como son los espejos, lentes, instrumentos, etc. Por ello, y aprovechando que la cúpula es metálica, cada caja es fijada a la cúpula mediante cuatro imanes de neodimio capaces de aguantar 2’5kg cada uno de ellos, algo absolutamente desmesurado pero que proporciona mucha seguridad.
Finalmente indicar que el proceso de impresión de cada caja ocupa aproximadamente dos días y que, al haberse utilizado filamentos de diferentes colores en la fabricación de las diferentes cajas, todas ellas han sido pintadas de negro por uniformidad y para evitar posibles reflejos.





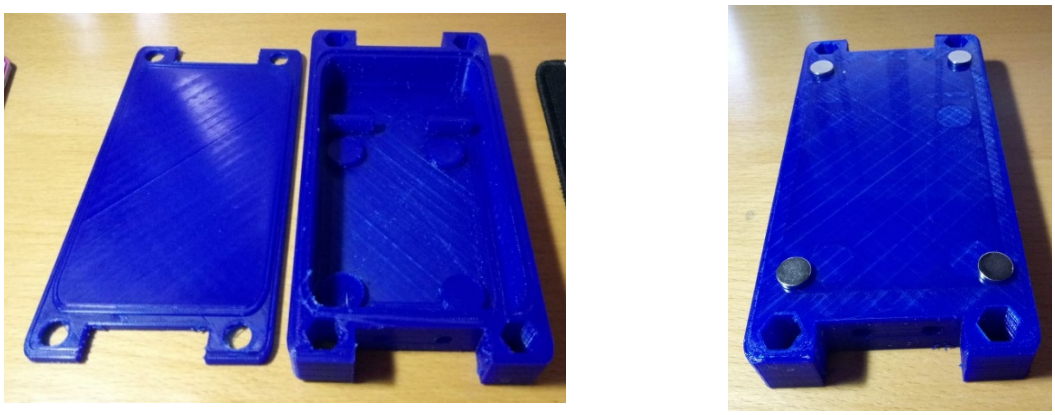


Ilustración 30. Caja diseñada para alojar la electrónica. Superior izquierda, visión lateral de la caja. Superior derecha, se observan los orificios para salida de dos termistores e interruptor. Inferior izquierda, el interior de la caja con el compartimento para alojar la pila de 9v, los soportes de la PCB y otros dos orificios frontales para la salida de los termisores. Inferior derecha, parte posterior de la caja con sus 4 imanes de neodimio para fijar la caja a la cúpula de metal. Tras la fabricación todas las cajas se han pintado de negro para evitar brillos y mejorar la uniformidad

5.1.7 Esquema físico

A continuación, se muestra un esquema muy breve de todos los componentes implicados a nivel físico en el presente proyecto

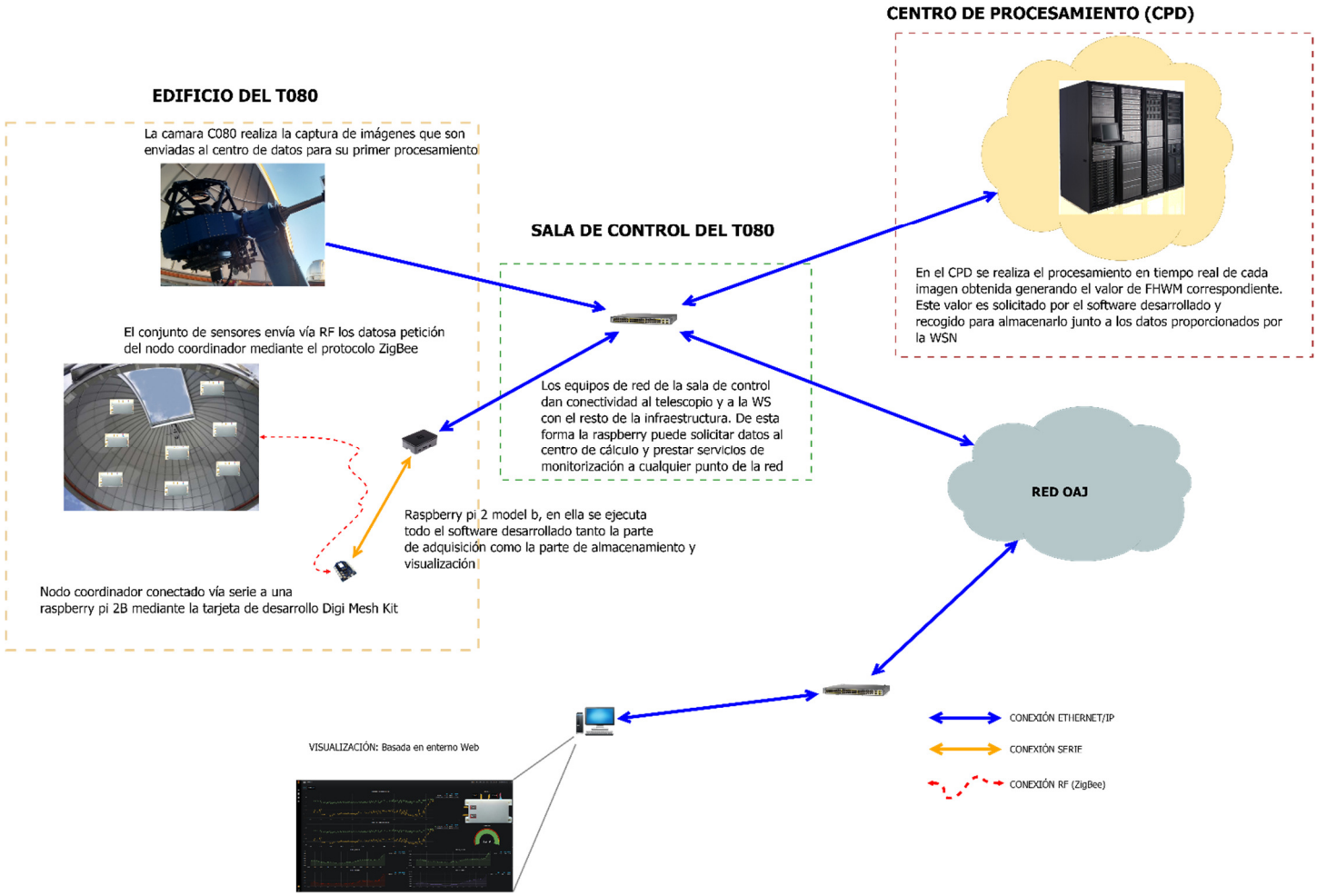


Ilustración 31. Esquema físico de todos los componentes que forman el sistema y su relación. La red de sensores se sitúa en el edificio del T080 y recibe conectividad con la red del OAJ mediante el equipamiento de la sala de control del T080. El equipo de adquisición también solicita el cálculo de la FWHM de las imágenes al

centro de procesamiento y el software de monitorización es accesible desde cualquier punto de la red (siempre que se cumplan las políticas de seguridad del OAJ)

Por otra parte, la secuenciación temporal que sigue todo el proceso es la siguiente:

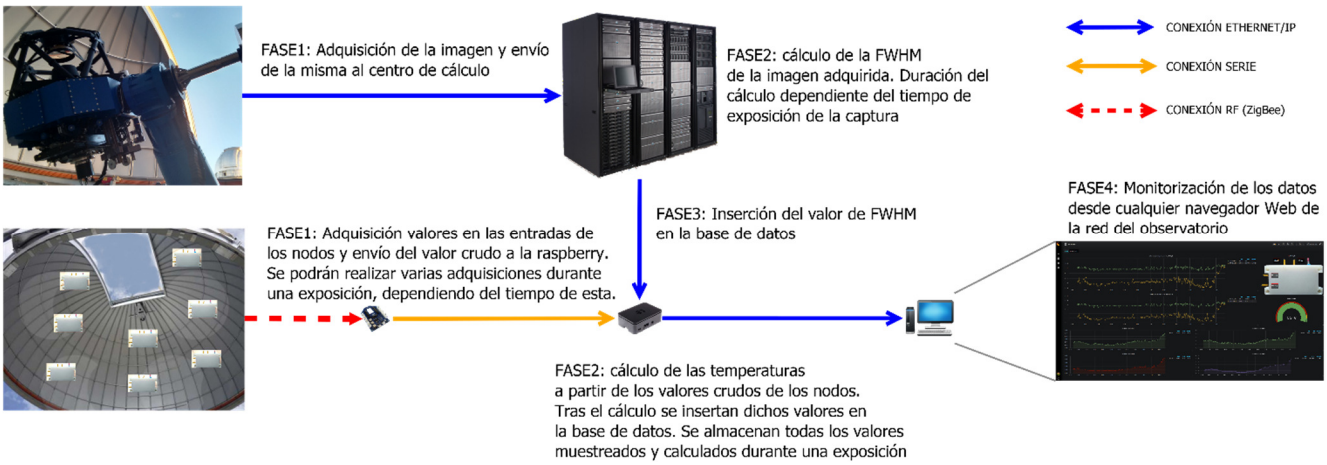


Ilustración 32. Secuenciación temporal del proceso desde la adquisición hasta la monitorización. En primer lugar, se adquieren las imágenes y los datos de temperatura de forma simultánea. En segundo lugar, el centro de procesamiento calcula la FHWM de la imagen obtenida. En la fase tres, dicho valor de FHWM es recogido por el sistema de adquisición desarrollado en este TFM para incluirlo como métrica en la base de datos. Finalmente, los datos están disponibles para su monitorización.

5.2 Software

En los siguientes apartados se va a explicar cuál ha sido el proceso para la elaboración del software que proporciona la funcionalidad requerida, además se hablará de las herramientas utilizadas y se justificará su uso. Hay que destacar también, que en este apartado no se explicará ni la instalación del software ni el manejo del programa, dado que esto se incorpora en los anexos *("12.3. Manual básico de instalación" y "12.4. Manual de usuario")*.

5.2.1 Entorno de desarrollo nodos WSN (radio)

Tal y como se ha ido comentando a lo largo de la presente memoria, se hace uso del *Digi Mesh Kit* para las pruebas iniciales, como decíamos incluye todas las herramientas hardware necesarias para su programación a través del software que proporciona Digi y que comentamos a continuación.

5.2.1.1 Digi XCTU

XCTU [81] es el software que Digi facilita gratuitamente, desde su página Web, para la programación de los módulos XBee, cuando decimos programación, realmente nos estamos refiriendo a la parametrización de los mismos, ya que como hemos comentado estos dispositivos permiten la modificación de una serie de parámetros que afectan a su comportamiento.

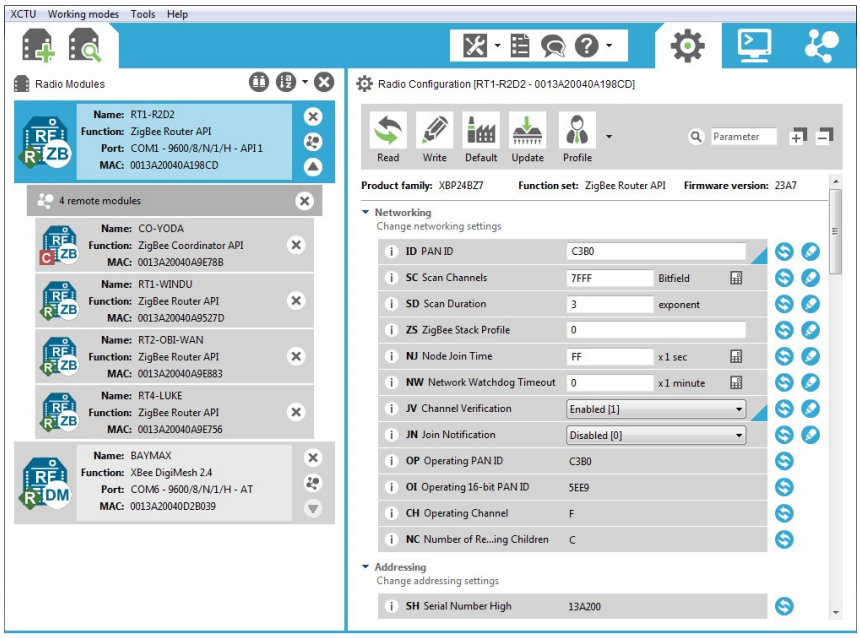


Ilustración 33. Software XCTU que Digi proporciona y que puede ser utilizado para la parametrización de los nodos XBee [81]

En el momento de la redacción del presente documento (mayo 2018) la versión, del software es la 6.3.13, es compatible con sistemas Windows o Unix/Linux y las características más relevantes que posee son:

- Permite el descubrimiento tanto de nodos locales (conectados vía serie) como de los nodos remotos (conectados vía radio).
- Permite la modificación/lectura de todos los parámetros de configuración de los módulos.
- Permite restaurar los nodos a fábrica y actualizar/reinstalar su firmware.
- Permite el envío de comandos y datos entre nodos tanto en modo transparente, mediante una consola de puerto serie, como en modo API (encapsulados en un *frame*) mediante un generador de *frames*.
- Es posible realizar un escaneo de la red y genera una visualización.

5.2.2 Gateway nodo colector. Parte comunicación

A continuación, se dará una explicación de todo el software desarrollado y de los motivos de la elección de las distintas herramientas, no obstante, antes de empezar, a modo de resumen general, se puede decir que, a nivel funcional, el software está dividido en dos grandes bloques.
Un primer bloque basado en Python 3 que realiza las tareas de gestión y configuración de los nodos, por tanto, de la red, y que además realiza la adquisición periódica de los valores de temperatura de los cuatro sensores de cada uno de los nueve nodos y la del valor de FWHM del centro de cálculo.
Por su parte, el segundo bloque se refiere al sistema de almacenamiento y visualización de datos, aunque se hablará de ello con más detalle, el software de almacenamiento está preparado para capturar los datos de todas las fuentes e introducirlo en una base de datos, por otra parte, el sistema de visualización desarrollado se basa en herramientas de código abierto y entorno Web.

Finalmente indicar, tal y como se ha mencionado, que aunque el software desarrollado es multiplataforma, el sistema operativo instalado en la *raspberry pi* es Ubuntu 16.0.4.4 LTS en su versión mínima al que se le han instalado el servidor X [82] para poder lanzar el interface gráfico, Aunque se comentará con más detalle en los siguientes puntos y anexos el software instalado es:

- Python 3 [83].
- InfluxDB [84].
- Telegraf [84].
- Grafana [85].

5.2.2.1 Python 3 (Pycharm)

A la hora de elegir el lenguaje para desarrollar la aplicación era imprescindible determinar si Digi proporcionaba de forma oficial ejemplos, código, librerías, etc.
Tras leer la documentación y visitando los foros, se comprueba que Digi ofrece una serie de librerías oficiales para acceder a los módulos y otras que, aun siendo desarrolladas por terceros, también son recogidas, comentadas y enlazadas por Digi [72].

- Oficiales:
  - Librería basada en Java.
  - Librería basada en Python.
  - Librería basada en ANSI C.
- No oficiales:
  - Librería basada en Arduino.
  - Node.js.
  - Todo tipo de ejemplos desarrollados por la comunidad y accesibles desde la Web de Digi.

Una vez determinados que lenguajes eran válidos y viendo que las librerías oficiales incorporaban todos los módulos necesarios para realizar la aplicación prevista, nos decantamos por la utilización de la librería Python. Los motivos de esta decisión son absolutamente personales y se basan en que el lenguaje Python es mayoritario en los entornos científicos y, por tanto, también lo es en el observatorio, eso es así en el ámbito científico, pero no tan habitual en tareas de ingeniería, por tanto, resultaba muy interesante iniciarse en este lenguaje a través del presente proyecto.

La librería está basada en Python 3 y por ello se elige estar versión del programa, por otra parte, para facilitar la gestión de archivos y desarrollo se elige el entorno *PyCharm* [86] al ser ampliamente respaldado, poseer una versión comunitaria y estar disponible para Windows, *macOs* y Linux.

Antes de continuar, hay que indicar que la parte Python del proyecto se encarga de tres bloques de tareas fundamentales:

- La gestión y configuración de los nodos y, por tanto de la red
- Las tareas de adquisición de temperaturas mediante peticiones periódicas a los nodos por parte del nodo coordinador.
- La petición al centro de cálculo de los valores de FWHM calculados para las imágenes científicas.

Atendiendo al bloque configurador y de gestión, cabe decir que su elaboración atendía a la necesidad de probar la librería, por tanto, se decidió programar todas las tareas que el software XCTU, del que ya hemos hablado, provee y van a ser necesarias para crear nuestra red de sensores, esto ha hecho que el proyecto sea absolutamente autónomo y no sea necesaria la utilización de otro software para la gestión plena de la red.

Los módulos Python programados, en la parte de configuración y monitorización son los siguientes:

- ***discover_devices_xbee.py:***
  Este módulo permite descubrir todos los nodos activos que pertenecen a la misma red que el nodo coordinador.
- ***read_local_params_xbee.py:***
  Realiza la lectura de parámetros del nodo local a través del puerto serie, es por ello por lo que debe leer ciertos parámetros de configuración, como nombre del puerto serie, velocidad puerto desde un archivo de configuración *config/sys_config.ini.*
- ***read_remote_parmas_xbee.py:***
  Al igual que el módulo anterior, obtiene los parámetros de configuración de un nodo, pero en este caso al tratarse de un modo remoto, se debe indicar la dirección del nodo a leer.
- ***write_local_params_xbee.py:***
  Realiza la escritura de parámetros del nodo local a través del puerto serie, para ello se leen los parámetros de configuración desde el archivo *local_node.ini.*
- ***write_remote_parmas_xbee.py:***
  Escribe los parámetros de configuración de un nodo, pero en este caso se trata de un modo remoto, se debe indicar la dirección del nodo a leer y al igual que el anterior el nodo debe poseer un archivo nombrado con dicha dirección en la que vengan recogidos todos sus parámetros. *Ejemplo de nombre válido config/0013S200416299FE.ini.*
- ***gui.py:***
  Es el módulo principal para lanzar la interface gráfica desde la cual el usuario podrá invocar al resto de módulos para realizar todas las tareas de configuración y diagnóstico. Esta interfaz gráfica se apoya en la librería gráfica de Python llamada *Tkinker* [87], se ha escogido esta librería porque es de facto la librería estándar de Python dado que se incluye en su distribución base. De esta forma no se hace necesaria la instalación de nuevos componentes para desarrollar y ejecutar un entorno gráfico

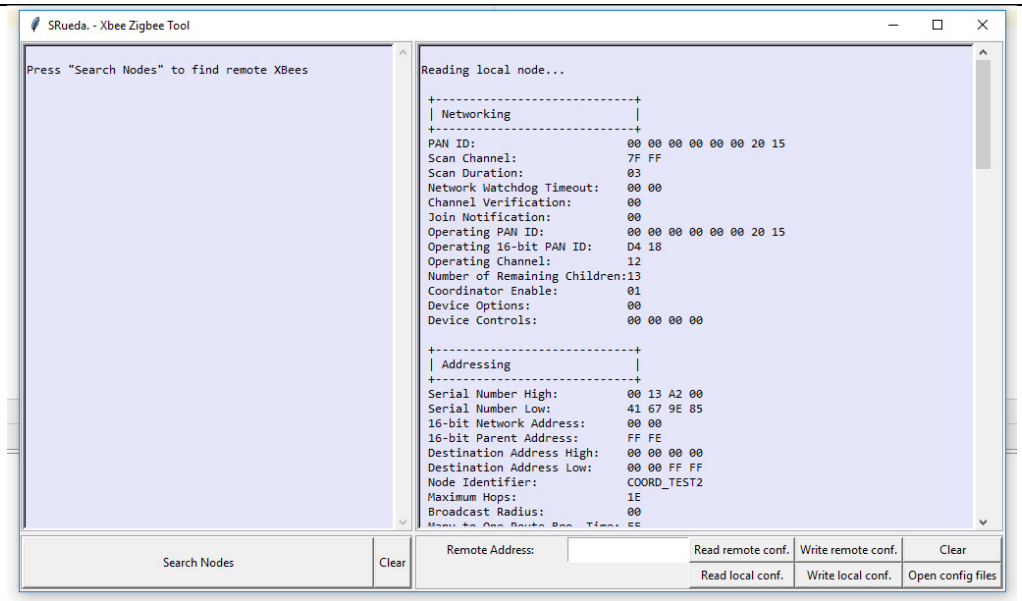


Ilustración 34. GUI desarrollado para la el descubrimiento de los nodos dentro de la red y la configuración tanto en local como en remoto de dichos nodos. En la imagen se muestra la lectura de parámetros del nodo local

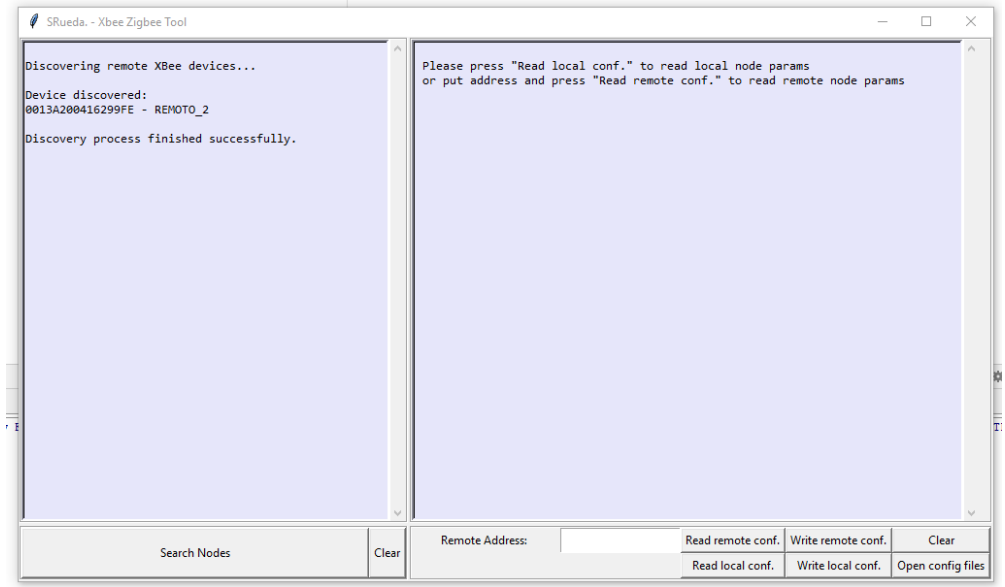


Ilustración 35. GUI desarrollado en el que se ha lanzado el descubrimiento de los nodos remotos dentro de la red. En la imagen se aprecia cómo sólo un nodo "REMOTO_2" está accesible.

Además de dichos módulos también se han programado otros que realizan tareas auxiliares y que son invocados por los anteriores:

- ***read_node_config_file.py:***
  Incorpora funciones de lectura de cada parámetro de un nodo desde los ficheros de configuración, como se ha dicho ese archivo es de tipo ini y su nombre coincide con su dirección para los nodos remotos y simplemente local_node.ini para el nodo que se accede por el puerto serie.

- ***read_node_list.py:***
  En un archivo de configuración, list_of_nodes.ini, indicamos que nodos de la red deben ser encuestados (existe la posibilidad de que no queramos que un nodo, aun perteneciendo a la red, sea interrogado), por tanto, desde este módulo se añade la funcionalidad de leer el fichero en el que se indican los módulos a interrogar para obtener sus valores de temperatura.
- ***read_sys_config.py:***
  Permite la lectura del archivo de configuración de parámetros del puerto serie.

Finalmente existen otros módulos que no son lanzados desde la interface gráfica de configuración, sino desde la línea de comandos y que permiten realizar las tareas de adquisición y almacenamiento de datos.

- ***read_remote_ADC_xbee_multihilo.py:***
  Este módulo realiza encuestas periódicas a todos los nodos sobre el estado de sus entradas analógicas y convierte los valores crudos a temperatura, el hecho de que sea multihilo permite lanzar la adquisición de cada nodo a una frecuencia de muestreo diferente y que esta sea independiente de las del resto. Esto permite también la gestión de errores y que la desconexión de nodos en fallo no afecte a todo el programa, ya que bastará con finalizar el hilo del nodo que esté originando los problemas. No obstante, debido a que el acceso a todos los nodos se tiene que hacer mediante el puerto serie, hay que implementar técnicas de bloqueo y desbloqueo de hilos.
  A continuación, se muestra el *log* que lanza el programa para una red de tres nodos, (un hilo para cada nodo, más los hilos principales) en el que se puede observar la fase de descubrimiento de nodos en la red y las posteriores peticiones a cada uno de ellos con los correspondientes bloqueos de recursos, el nombre del hilo se muestra al inicio de cada línea entre paréntesis *(parte del log se ha eliminado para aportar claridad al ejemplo)*:

```
Apertura del puerto serie e inicio del proceso de descubrimiento:
(MainThread) COM6 port opened
(MainThread) COM6  SENT       OperatingMode.API_MODE7E 00 04 08 01 41 50 65
(Thread-6  ) COM6  RECEIVED   OperatingMode.API_MODE7E 00 06 88 01 41 50 00 02 E3
Descubrimiento del nodo 1:
 (MainThread) COM6  SENT       OperatingMode.ESCAPED_API_MODE7E 00 0C 08 0A 4E 44 52 45 4D 4F 54 4F 5F 31 F5
(Thread-6  ) COM6  RECEIVED   OperatingMode.ESCAPED_API_MODE7E 00 20 88 0A 4E 44 00 97 BD 00 13 A2 00 41 62 9A
0E 52 45 4D 4F 54 4F 5F 31 00 00 00 02 00 C1 05 10 1E 2B
(MainThread) Descubierto: 0013A20041629A0E - REMOTO_1
Descubrimiento del nodo 2:
 (MainThread) COM6  SENT       OperatingMode.ESCAPED_API_MODE7E 00 0C 08 0D 4E 44 52 45 4D 4F 54 4F 5F 32 F1
(Thread-6  ) COM6  RECEIVED   OperatingMode.ESCAPED_API_MODE7E 00 20 88 0D 4E 44 00 DC 3A 00 13 A2 00 41 62 99
FE 52 45 4D 4F 54 4F 5F 32 00 00 00 02 00 C1 05 10 1E 76
(MainThread) Descubierto: 0013A200416299FE - REMOTO_2
Descubrimiento del nodo 3:
(MainThread) COM6  SENT       OperatingMode.ESCAPED_API_MODE7E 00 0C 08 10 4E 44 52 45 4D 4F 54 4F 5F 33 ED
(Thread-6  ) COM6  RECEIVED   OperatingMode.ESCAPED_API_MODE7E 00 20 88 10 4E 44 00 16 48 00 13 A2 00 41 62 9A
08 52 45 4D 4F 54 4F 5F 33 00 00 00 02 00 C1 05 10 1E 1F
(MainThread) Descubierto: 0013A20041629A08 - REMOTO_3
Crea hilo para el nodo 1 (0013A20041629A0E - REMOTO_1:
(MainThread) creando hilo
(0013A20041629A0E - REMOTO_1) Stamp: 2018-05-22 19:46:12.164251
```

(0013A20041629A0E - REMOTO_1) Waiting for lock

Bloquea recursos del puerto para el nodo 1, esto significa que ya ha pedido los valores de temperatura para este nodo

(0013A20041629A0E - REMOTO_1) Acquired lock

Crea hilo para el nodo 2 (0013A200416299FE - REMOTO_2):

(MainThread) creando hilo

(0013A200416299FE - REMOTO_2) Stamp: 2018-05-22 19:46:12.174225

Espera a que el recurso del puerto esté libre para hace la petición al nodo de sus temperaturas

(0013A200416299FE - REMOTO_2) Waiting for lock

Crea hilo para el nodo 3 (0013A20041629A08 - REMOTO_3):

(MainThread) creando hilo

(0013A20041629A08 - REMOTO_3) Stamp: 2018-05-22 19:46:12.178213

Espera a que el recurso del puerto esté libre para hace la petición al nodo de sus temperaturas

(0013A20041629A08 - REMOTO_3) Waiting for lock

Recibe temperaturas a través del hilo de REMOTO-1

(0013A20041629A0E - REMOTO_1) ntc 10K temper: 23.76

(0013A20041629A0E - REMOTO_1) ntc 10K temper: 23.70

(0013A20041629A0E - REMOTO_1) ntc 10K temper: 23.78

(0013A20041629A0E - REMOTO_1) ntc 10K temper: 23.76

El hilo de REMOTO-1 libera recursos y espera hasta que tenga que volver a pedir valores al nodo REMOTO_1

(0013A20041629A0E - REMOTO_1) Sleeping 30.00

Bloquea recursos del puerto para el nodo 2, esto significa que ya ha pedido los valores de temperatura para este nodo

(0013A200416299FE - REMOTO_2) Acquired lock

Recibe temperaturas a través del hilo de REMOTO-2

(0013A200416299FE - REMOTO_2) ntc 10K temper: 24.12

(0013A200416299FE - REMOTO_2) ntc 10K temper: 24.22

(0013A200416299FE - REMOTO_2) ntc 10K temper: 24.20

(0013A200416299FE - REMOTO_2) ntc 10K temper: 24.22

El hilo de REMOTO-2 libera recursos y espera hasta que tenga que volver a pedir valores al nodo REMOTO_2

(0013A200416299FE - REMOTO_2) Sleeping 30.00

Bloquea recursos del puerto para el nodo 3, esto significa que ya ha pedido los valores de temperatura para este nodo

(0013A20041629A08 - REMOTO_3) Acquired lock

Recibe temperaturas a través del hilo de REMOTO-3

(0013A20041629A08 - REMOTO_3) ntc 10K temper: 23.94

(0013A20041629A08 - REMOTO_3) ntc 10K temper: 23.59

(0013A20041629A08 - REMOTO_3) ntc 10K temper: 24.11

(0013A20041629A08 - REMOTO_3) ntc 10K temper: 24.29

(0013A20041629A08 - REMOTO_3) Sleeping 30.00

El proceso anterior se repite en un bucle infinito añadiendo hilos según el número de nodos que queramos añadir a la red, en este bucle se va solicitando a cada nodo el valor de sus entradas cada 30 segundos (es un valor configurable y que puede ser independiente para cada nodo) Por otra parte si un nodo entra en fallo y no envía su respuesta, su hilo se acelera para encuestarlo cada segundo (valor también configurable), con la intención de conocer si ha sido un problema temporal u otro permanente como por ejemplo, el agotamiento de la batería, en caso de que se generen 20 fallos seguidos en un nodo (este valor también es modificable) se considera que el nodo está en error irreversible sin actuación humana con lo que se libera su hilo y se deja de encuestar al nodo en cuestión.

- ***get_fwhm.py:***
  Nos permite obtener la FWHM de las imágenes obtenidas a partir de un tiempo determinado, este tiempo es configurable a través de una constante en el código.
  Para obtener esta FWHM lo que se hace es una consulta a la base de datos del centro de cálculo que almacena diferente información de las imágenes astronómicas capturadas, también se aprovecha que uno de los datos que se almacenan es el *timestamp* de cuándo se realizó la adquisición de esta imagen, lo que nos permite una ejecución asíncrona de este módulo, es decir, independientemente del momento en que lo ejecutemos, el módulo recogerá al FWHM de cada imagen y la fecha en que se realizó, de forma que en nuestro sistema se registrará el valor de FWHM con la fecha exacta en la que se produjo la adquisición (año/mes/día hora:minutos:segundos).
  send_data_to_telegraf.py:
  Este módulo es llamado por los tres anteriores para que haga la inserción de los valores obtenidos en la base de datos, aunque a continuación, se hablará en más detalle del sistema de almacenamiento, cabe decir que se ha instalado un servicio intermedio llamado *telegraf* que realiza las tareas de Gateway entre los generadores de datos y el sistema de almacenamiento.

A continuación se listan las librerías de Python que no se incluyen por defecto en la instalación base y que requerirán ser instaladas manualmente, el modo de instalación se comentará en los anexos (*"12.3 Manual básico de instalación")*, aunque a modo de resumen podemos indicar que la forma más sencilla de realizarla es a través de la herramienta pip3 [88] aunque también serán proporcionados en este TFM para su instalación manual.

- **pyserial-3.4:** Permite el acceso al puerto serie del equipo
- **pytelegraf-0.3.1:** Necesario para enviar las mediciones al sistema de recolección, se hablará de ello más adelante
- **digi_xbee-1.1.0:** Se descarga desde la Web de Digi e incluye todas las funciones para establecer comunicación con los dispositivos XBee

Para finalizar, indicar que el software también incluye una serie de archivos de configuración que nos permitirán la correcta ejecución del programa y su adaptación a cambios en el host colector o en alguno de los nodos

- **sys_config.ini:** Indica parámetros del puerto serie, como nombre y velocidad.
- **local_node.ini:** Incluye todos los parámetros del dispositivo que queremos modificar por modo local, recordemos que a este nodo se accede por el puerto serie, por tanto, no es necesario conocer su dirección ya que ninguna tarea de RF es necesaria.
- ***DirecciónDelNodo.ini*** *(ejemplo:* **0013A200415BFC59.ini):** Incluye todos los parámetros del dispositivo que queremos modificar vía RF, cada vez que queramos configurar un nodo remoto basta con editar o crear un archivo de tipo *ini* y nombrado con su dirección que contenga todos sus parámetros.
- **List_of_nodes.ini:** En este archivo se listan los identificadores de los nodos, esto es los NI (es una cadena de texto que identifica a los nodos, del inglés, Node *Identifier*) de los nodos que formando parte de la red queremos que sean

interrogados por el nodo colector. Esto nos permite ir añadiendo o eliminando nodos según la necesidad.

5.2.3 Gateway nodo colector. Tratamiento de datos

Uno de los aspectos a el que también se ha querido prestar especial atención, es al sistema de recolección y visualización de datos, la idea era proporcionar métodos flexibles que permitieran la recolección de información de un conjunto heterogéneo de fuentes, que a su vez eran muy dinámicas y variables, y, por otro lado, que posibilitaran la visualización de la información desde un entorno Web, siendo, por tanto, accesibles de forma concurrente y desde distintos tipos de dispositivos.
Notar, por tanto, que no se ha pretendido generar un sistema de tratamiento de datos desde cero, sino que se ha buscado la adaptación de herramientas de código abierto a los objetivos de este TFM.
Antes de comenzar con la descripción de los componentes utilizados, indicar que, por comodidad, todos ellos se han instalado en la *rapsberry pi,* no obstante, su instalación total o parcial podría hacerse en cualquier otro equipo de la red dado que el acceso y la comunicación entre ellos se realiza mediante protocolos *http*, *udp* y/o *tcp*.
Finalmente indicar que el nodo colector sincroniza su reloj interno mediante los servidores NTP del observatorio (NTP del inglés, *Network Time protocol*, es un protocolo de Internet para sincronizar los relojes de los sistemas informáticos a través del enrutamiento de paquetes en redes con latencia variable. Se basa en un sistema cliente-servidor y tiene una estructura jerárquica, ya que existe un servidor primario de referencia que se asienta en el más alto nivel de la jerarquía. Este servidor primario está seguido de servidores secundarios de referencia y clientes) de esta manera trabaja en el mismo horario que el centro de cálculo y que en los observatorios es siempre UTC.

5.2.3.1 InfluxDB

InfluxDB [84] es una base de código abierto y multiplataforma pensada para el almacenamiento de datos de series temporales, esto quiere decir que los datos que se le envían son marcados con su tiempo de inserción y, por tanto, almacenados en ese orden. Podemos decir que sirven para medir el valor de una variable a lo largo del tiempo y por tanto ideal para operaciones de monitorización.
A la hora de diseñar una base de datos no hay que hacer una previsión de las columnas que va a tener, simplemente se van añadiendo al vuelo a medida que vamos añadiendo métricas a historizar.
En el presente proyecto se instaló la versión 1.5.2 de *influxDB*.

5.2.3.2 Telegraf

Telegraf es un *plugin* o agente ligero de *influxDB* destinado a la recolección de datos también llamados métricas, su principal propósito es la de servir de *gateway* entre fuentes heterogéneas y la base de datos.
Para ello *telegraf* incorpora una gran variedad de sistemas de entrada, apache, *http post, fluentd, udp, udp6, tcp, tcp6,* etc. Y una gran variedad de conexiones de salida.

Todo esto se moldea mediante la modificación de un archivo de configuración (*telegraf.conf*).

En nuestro caso debemos señalar tres aspectos:

- Como fuente de entrada se ha activado la conexión vía *udp*, con esto le estamos diciendo al agente que va a recibir entradas mediante envíos *udp*, que en nuestro caso serán generadas por el código *Python*: *send_data_to_telegraf.py.*
- Como destino de salida se configura en el mismo archivo la conexión contra *influxDB*, simplemente indicando la dirección y puerto de la base de datos.
- Por último*, telegraf* debe recoger los datos en un formato específico, para evitar tener que realizar un elaborado formateo de los mismos, es decir, de las métricas a enviar. Se ha utilizado la librería *Pytelegraf* [89] que permite el envío de los valores mediante el uso de funciones simples.

5.2.3.3 Grafana

Grafana [85] es una herramienta de código abierto para el análisis y visualización de métricas vía web. Hay tres versiones de *grafana* accesibles hoy en día, una versión libre para descargar e instalar en nuestra propia máquina que es la que nosotros utilizamos, otra basada en la nube con planes gratuitos y de pago y, finalmente, una versión Enterprise de pago que incluye soporte, *plugins* adicionales, cursos, etc.
Dentro de sus ventajas están:

- Permite multitud de fuentes de datos basadas en series de tiempos. Así una vez configurado dicho origen de datos incorpora un editor de *queries* adaptado y customizado para las particularidades de dicha fuente.
- Permite el diseño de paneles de datos (*dashboards*) para múltiples organizaciones, con lo que cada organización puede tener sus propios paneles con variados orígenes de datos.
- En cada panel se pueden incorporar diferentes tipos de gráficas, como lineales, tablas, calibres, etc.
- Proporciona herramientas para la gestión de usuarios, de forma que estos usuarios pueden ser añadidos al sistema dándoles distintos permisos según el rol que vayan a desempeñar.
- Es posible crear alarmas cuando alguna de las métricas se sitúe fuera de los márgenes indicados.
- Permite un ajuste muy flexible del periodo de visualización desde el propio *dashboard.*
- Admite la instalación de *plugins* para añadir funcionalidades a la visualización, la instalación base ya incorpora una gran cantidad de ellos, se ha añadido también el plugin *PictureIt* [85], descargable desde la propia Web de grafana, que permite la incorporación de archivos de imágenes en los paneles.

Ilustración 36. Interface Web desarrollado con Grafana del sistema de monitorización de un nodo. Se puede observar que se monitoriza el nodo "REMOTO_2", el nodo a monitorizar es seleccionable desde el desplegable situado en la parte superior izquierda. En la gráfica superior (amarilla y verde) se muestra la diferencia de temperatura entre pares de sensores (1vs2 y 3vs4), justo debajo, en azul, el valor de FWHM de las imágenes. Las cuatro gráficas inferiores muestran los valores históricos de los cuatro sensores que componen el nodo. Todas las gráficas incluyen el valor actual, el máximo y el mínimo. Finalmente, a la derecha se muestran una imagen del nodo con los valores actuales de temperatura de sus cuatro sensores y el valor de voltaje de la tensión de alimentación.

Comentar que la versión instalada es la v5.1.0 y el acceso a los paneles se hace desde cualquier navegador Web indicando la dirección del servidor configurado y el puerto 3000 (Por motivos de seguridad el acceso a las redes del observatorio desde una red exterior sólo es posible mediante el uso de la red privada virtual, *VPN, del inglés Virtual Private Network*, y la expedición del certificado de usuario correspondiente)

5.2.4 Esquema conceptual del conjunto

A continuación, se muestra un esquema en el que se detallan las relaciones lógicas que se establecen entre todos los componentes que forman parte del proyecto.

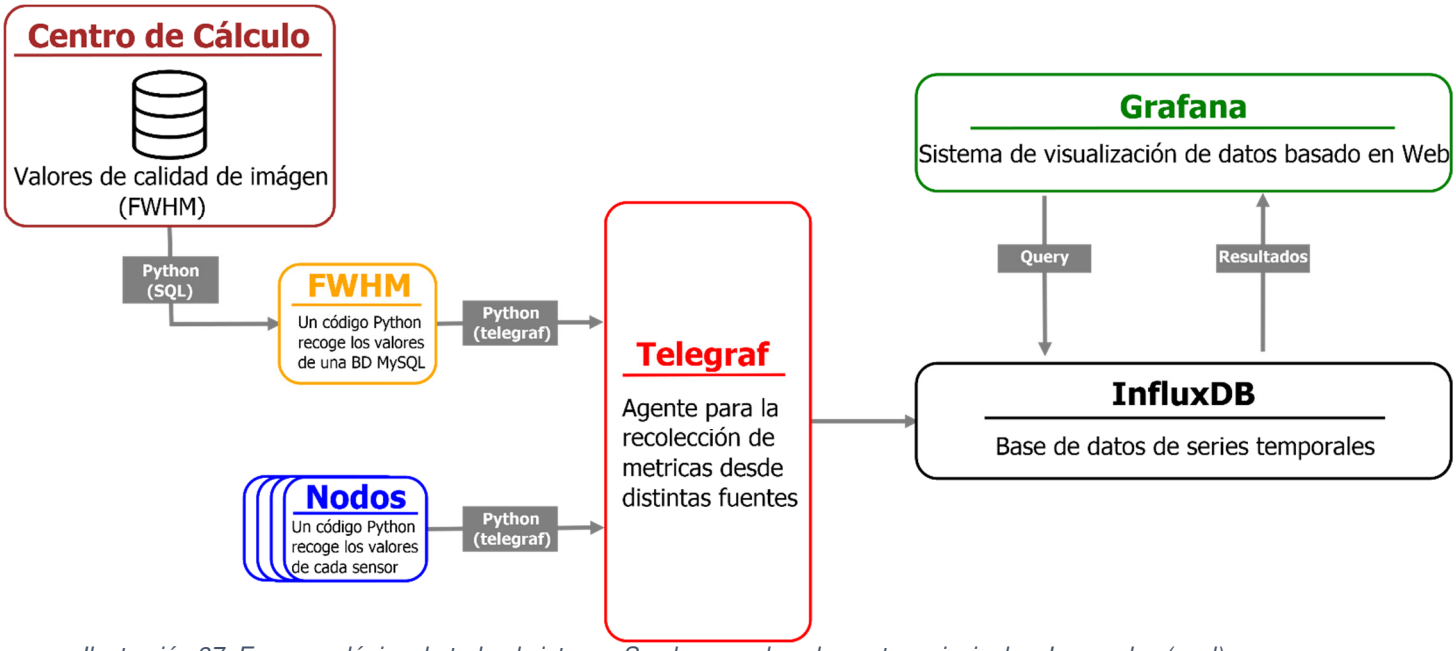


Ilustración 37. Esquema lógico de todo el sistema. Se observan los elementos principales. Los nodos (azul) con los que se capturan los datos de temperatura. El centro de cálculo (granate) que genera el valor de FWHM de las imágenes científicas, el software Python (amarillo) que se comunica con el centro de cálculo para obtener dicho valor FWHM. El agente Telegraf (rojo) que sirve de interfase entre las fuentes de datos heterogéneas y la base de datos. La propia base de datos InfluxDB (negro) donde se almacenan las distintas métricas, y finalmente el sistema de monitorización basado en web, Grafana (verde)

5.2.5 Diseño de las PCB. EasyEDA

Para el diseño de la placa de circuito impreso se ha utilizado el software EasyEDA [90], se trata de una aplicación para la simulación online de circuitos y diseño de PCBs, presenta la gran ventaja de que su uso es gratuito y completamente online y tendremos todas las funcionalidades simplemente con registrarnos en su web sin la necesidad de instalar ningún otro software.
Otro detalle muy importante es que, desde el propio programa, una vez concluido el diseño, podemos solicitar su fabricación desde la web, a unos precios y tiempos de provisión francamente buenos.

5.2.6 Software de diseño 3D de la caja. Tinkercad

Para el diseño de la caja que alberga la electrónica también se ha optado por la utilización de un software gratuito basado en la nube, se trata de Tinkercad [91] creado por Autodesk una de las empresas líderes en el diseño de diseño 3D.
El objetivo de Tinkercad, por tanto, es ofrecer una herramienta online de diseño e impresión 3D simple y destinada a todos los públicos independientemente de su nivel de conocimientos, esta sencillez en ocasiones choca cuando se necesitan herramientas para diseños más complejos, no obstante, ha cumplido de sobra con los requerimientos de nuestro proyecto.

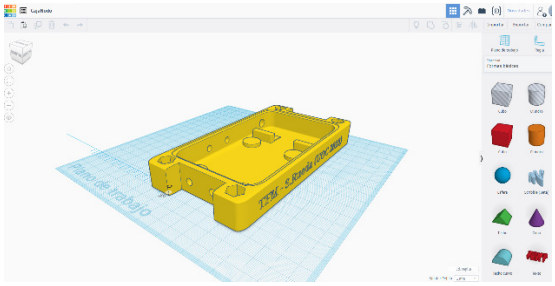


Ilustración 38. Diseño de la caja para la electrónica. Tinkercad

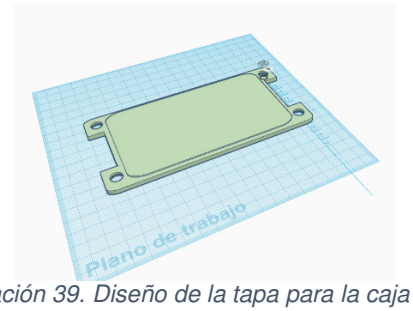


Ilustración 39. Diseño de la tapa para la caja de la electrónica. Tinkercad

Comentar que desde este programa, una vez acabado el diseño se pueden exportar ficheros en formato *stl*, estos ficheros deben ser importados por el software propietario de nuestra impresora 3D, (impresora *Leapfrog* BOLT PRO 3D), para generar los archivos *G-code* [92] que indicaran a la impresora las coordenadas sobre las que depositar el material.

6. Sistema prototipo

Lógicamente antes de comenzar con la producción “en serie” de los nodos se estudió la realización de varios prototipos con la idea de prever posibles problemas y puntos críticos. A continuación, se muestran los pasos que se siguieron en el desarrollo del software hasta llegar al producto final.

6.1 Pruebas de viabilidad en prototipo inicial.

Originalmente, una vez diseñado el circuito se trató de implementar en una *protoboard* para comprobar su viabilidad, sin embargo, se encontraron muchos problemas debido a lo sensible que es el conversor AD de los XBee, de forma que cualquier mínima variación en el contacto de alguna de sus entradas proporciona unos valores erróneos. Una vez comprobado que este camino no era fiable se decide realizar una versión simplificada del mismo en una placa pre-perforada, soldando los componentes, para determinar su viabilidad.

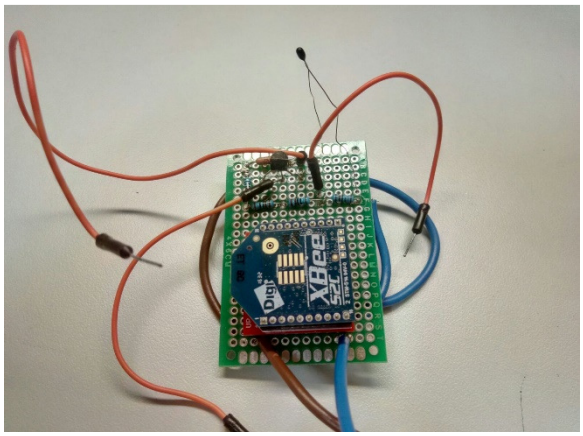


Ilustración 40. Primer prototipo de un nodo sensor realizado en placa pre-taladrada. Se utilizan tres entradas, una con termistor ntc, otra con un sensor LMT84 y otra con un simple divisor de tensión para validar los valores de las medidas

Como se puede apreciar en la imagen anterior en esta fase todavía no se había decido el tipo de sensor a elegir, por lo que se utilizaron tres entradas, una con un termistor, otra con un LMT84 y otra con un simple divisor de tensión para comprobar la precisión en la conversión del valor analógico. Los resultados fueron plenamente satisfactorios por lo que se decide enviar a fabricar las PCBs.

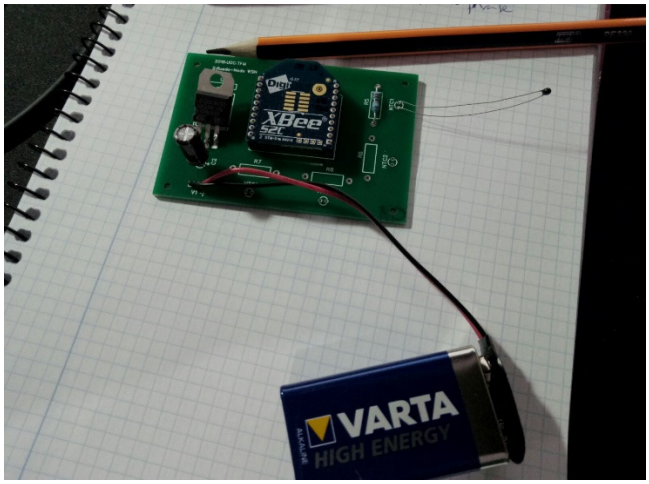


Ilustración 41. Prototipo final de un nodo sensor integrado en placa.

Tras las validaciones, todos los resultados se consideran más que aceptables por lo que se pasa a integrar todas las PCBs con sus componentes definitivos.

6.2 Integración en las PCBs

A continuación, se muestra una imagen de una placa totalmente acabada.

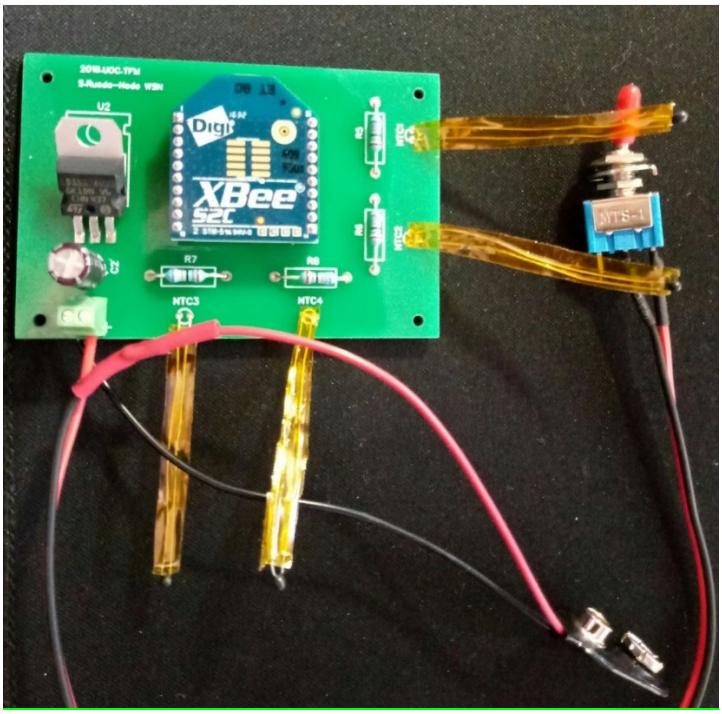


Ilustración 42 Fotografía de la PCB con todos los componentes integrados. Se cubren los termistores con cinta kapton para evitar cortocircuitos y proteger a dicho componentes ya que estarán más expuestos.

Tal y como se ve en la imagen, se decide cubrir los pines de los termistores con cinta *kapton* lo que protege el componente y evita cortocircuitos que puedan falsear la medida sin que por ello el sensor vea mermada su maleabilidad.

6.3 Pruebas de conjunto en laboratorio

Antes de la integración definitiva en el observatorio se prueba todo el sistema en el laboratorio para comprobar su correcto funcionamiento.

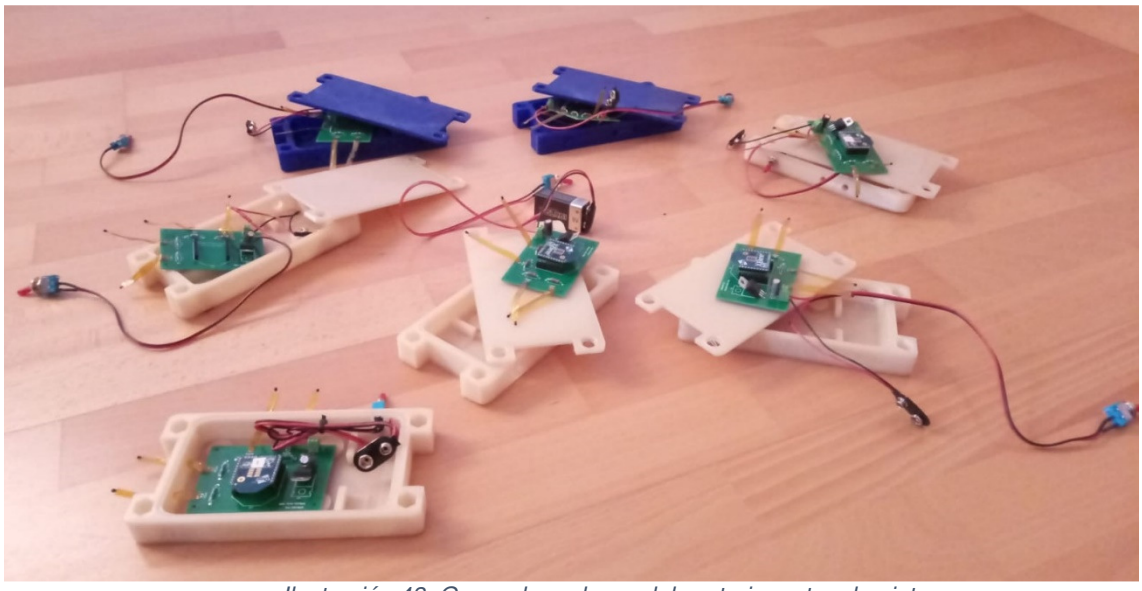


Ilustración 43. Grupo de nodos en laboratorio, antes de pintar

Tal y como se ha comentado y se puede apreciar en la imagen anterior, las cajas han sido elaboradas con distintos colores debido a que ya disponíamos de carretes de PLA de distintos colores, aunque después se pintan de negro.

Las pruebas que se han hecho en laboratorio han sido:

- Validación del alcance de la señal de radio: Se separaron los nodos más de 20m del nodo colector sin pérdidas de datos, lo cual se considera totalmente satisfactorio ya que no se espera una separación entre nodos finales y el nodo colector mayor de 10m dentro de la cúpula
- Validación del sistema de fijación por imanes: Se acoplaron los nodos a superficies metálicas y se sometieron a vibraciones y a tracciones sin que ningún nodo se desprendiese. Tal y como se indicó cada nodo va dotado de 4 imanes de neodimio que soportan más 2kg cada uno.
- Precisión en la medida: Los valores capturados por los nodos fueron contrastados con una estación meteorológica y con un termómetro por infrarrojos. A pesar de que no se observaron grandes discrepancias en las mediciones, las diferencias en las medidas oscilaban, según el sistema de adquisición empleado, entre 0'5ºC y 1ºC. Esto hace que esta prueba simplemente nos diese una idea orientativa de la validez del sistema desarrollado.
  Finamente, tal y como se ha dicho, se optó por un lado, por medir la tensión de entrada, lo que se realizó con un multímetro de buena calidad (Multímetro

digital Fluke FLUKE-87-V/EUR [93]) en todas las entradas comparando su valor con el mostrado por el software desarrollado, y por otro, repetir las mediciones colocando diferentes sensores en el mismo punto exacto sin que en ningún caso se apreciasen diferencias significativas (en alguna ocasión, como máximo del orden de centésimas de grado). Tal y como se ha dicho, a largo plazo resultaría interesante realizar una calibración más exhaustiva de los sensores, no obstante, se da este punto por validado, máxime si aún en el caso de que el error pueda existir, es un error del mismo orden en todos los nodos y en todos los sensores y por tanto no es significativo si trabajamos con diferencias entre ellos.

Indicar también que los nodos se han sometido a temperaturas que van desde los 3ºC (interior de una nevera) hasta los 37ºC con una correcta respuesta en la medida.

- Robustez del software: Se ha dejado el sistema funcionando con todos los nodos durante periodos de tiempo que oscilan desde un par de horas hasta tres días, y en su versión final se considera el software plenamente estable y listo para su puesta en producción.

A continuación, se muestran unas imágenes de los nodos terminados y listos para ser instalados en el observatorio:

Ilustración 44. Vista de todos los nodos finalizados. Como se observa todos los nodos remotos han sido pintados de negro y etiquetados para su identificación. Por otra parte, como nodo colector se usa uno de los adquiridos en el Digi-Mesh Kit conectado a una raspberry pi model b vía serie mediante cable USB y la placa de desarrollo correspondiente

7 Fase de despliegue

La fase de despliegue ha estado marcada, principalmente, por dos circunstancias. Una de ellas era conocida a priori, y es que el estudio del impacto que las variaciones de temperatura tienen en el *seeing* local es un proceso largo y laborioso que va a necesitar meses de recogida de datos y un análisis minucioso de los mismos, en la que se verán implicadas diversas disciplinas, como la ingeniería óptica, la mecánica, la electrónica y por su puesto el campo de la astrofísica, esto hace que el recorrido de este proyecto vaya más allá de la duración prevista para este TFM. La otra circunstancia tiene un carácter más aleatorio y es ajena a nuestro control, no se trata de otra cosa más que las condiciones meteorológicas y que, este año en particular, han sido inusualmente adversas, esto ha hecho que, desde que se acabaron las pruebas en laboratorio (penúltima semana de mayo) hasta la fecha actual (segunda semana de junio) prácticamente no hayamos podido disfrutar más que de unas pocas horas en las que se dieran las condiciones necesarias para la observación astronómica, por ello, aunque datos de temperatura sí se han podido recoger de forma continua y por tanto se ha podido validar la funcionalidad del sistema, estos datos no se han podido contrastar con los valores de FWHM al no haber imágenes astronómicas disponibles.
Teniendo en cuenta por tanto el poco tiempo de operación disponible se tomaron las siguientes decisiones:

- Se deja para más adelante el despliegue en el exterior de la cúpula.
- Dado que la determinación de la mejor ubicación de los nodos en el interior de la D080 tiene una fuerte componente empírica y serán necesarias diferentes pruebas se decide empezar por la zona próxima a la rendija de entrada de la cúpula ya que es la zona que está directamente en el camino óptico.

- Se van a muestrear datos en periodos de tiempo que oscilan entre los 30sg y 2min, si bien este tiempo es configurable.

Así pues, teniendo en cuenta las premisas anteriores se decide la siguiente ubicación de los sensores. (Por claridad la imagen muestra un esquema exterior de la cúpula, no obstante, la ubicación de los sensores se realiza en la posición marcada, pero en el interior de la misma)

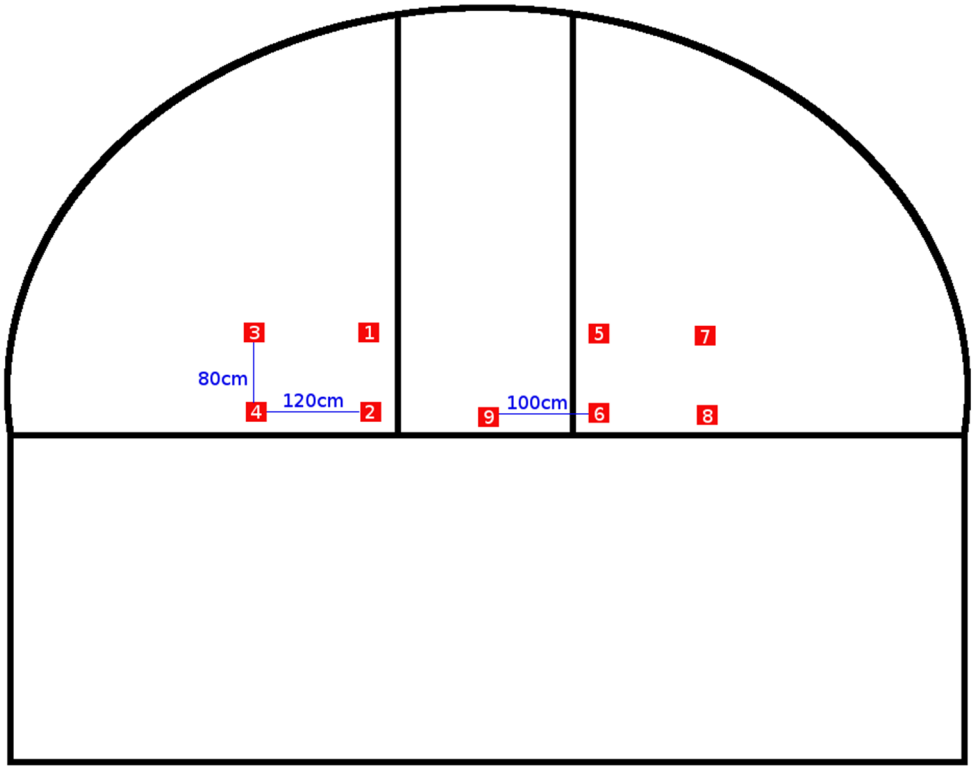


Ilustración 45. Esquema de colocación de los nodos en el interior de la D080. Como se observa se instalan cuatro nodos a cada lado de la rendija de entrada y uno en la posición central de la misma

A continuación, se muestran unas imágenes de la instalación real de dichos nodos en el interior de la cúpula:

Ilustración 46. Imágenes del despliegue real dentro de la D080. Se observan cuatro nodos a cada lado de la rendija de entrada y un nodo en la parte central-inferior de la misma

En cuanto a la ubicación del nodo colector, lógicamente al ser alimentado desde la red eléctrica (*raspberri pi*) se ha realizado en un cuadro eléctrico fijo de la instalación, por tanto, su posición respecto a los nodos es variable dependiendo del ángulo de rotación de la cúpula. La decisión de instalar la *raspberri* en el interior de un cuadro eléctrico viene marcada por la intención de proteger lo máximo posible todos los elementos que conforman el sistema.

Ilustración 47. En la imagen superior se muestran tres cuadros eléctricos fijos para el control de la cúpula. La raspberri pi se ha instalado en el cuadro inferior y el módulo XBee se sitúa sobre el cuadro izquierdo (no se introduce en ningún cuadro para maximizar el alcance de la señal RF. En la imagen inferior vemos abierta la caja dónde se ha ubicado la raspberri pi

Finalmente indicar que el sistema de monitorización es accesible desde cualquier puesto de control del observatorio y también desde cualquier ubicación remota mediante una red privada virtual (VPN) y el certificado de seguridad correspondiente.

Ilustración 48. Imagen del sistema de monitorización diseñado, desde uno de los puestos de la sala de control principal del observatorio. En la actualidad el observatorio cuenta con tres salas de control, una remota ubicada en la sede de Teruel, otra menor en el propio observatorio, principalmente destinada a el control del telescopio T080 y otra principal, también ubicada en el observatorio, desde la que se suelen controlar todos los telescopios y sistemas y uno de cuyos puestos aparece en estas imágenes

8 Valoración económica del trabajo

A continuación, se realiza un resumen económico que contiene tantos los productos adquiridos como una valoración de los trabajos de ingeniería. Los módulos de *XBee* deben ser importados y en la fecha de la realización de este proyecto, sólo existen dos proveedores Digi-Key Electronics [94] y Mouser [95], otro tipo de material menos específico se adquirió en Amazon o RS-Online. También indicar que, al ser los gastos mayores de 50€ en cada pedido, no se aplican gastos de envío para el cliente.

8.1 Kit de iniciación y prototipo

Identificador	Producto	PVPu	Cantidad	Total
Digi-Key: 602-1826-ND **Digi**: XKB2-Z7T-WZM	XBee Zigbee Mesh Kit	73'8€	1	73'8€
Digi-Key: 615-1069-ND **US Sensor:** PR103J2	Termistor NTC 10KOhms	5'31€	4	20'52€
Digi-Key: CMF210KHBCT-ND **Vishay Dale:** CMF55210K00BEEB	Resistencia 210KOhms 1/2W 0.1% Axial	0'76€	8	6'08€
Digi-Key: 615-1069-ND **Texas Instrument:** LMT84LP	sensor de temperatura Analógico, local -50 °C ~ 150 °C 5.5 mV/ °C TO-92-3	0'68€	4	2'72€
Digi-Key: S5751-10-ND **Sullins Connectors:** NPPN101BFCN-RC	Cabecera Conector 0.079" (2.00 mm)	0'8€	4	3'2€
			Total, sin impuestos:	**106'32€**
			Impuestos (21%):	**22'33€**
			Gastos de importación (20%):	**21'26€**
Amazon: B0723976PW	Pack de 4 baterías alcalinas 9v	7'95€ (IVA incl.)	**1**	**7'95**
	Material vario: cables, placas pre-taladradas, regulador, condensador, etc.	10€ (IVA incl.)	**1**	**10€**
PRECIO TOTAL DEL MATERIAL PARA PROTOTIPOS:				**167'86€**

8.2 Fabricación de nodos

Identificador	Producto	PVPu	Cantidad	Total
Mouser: 888-XB24CZ7PIT-004 **Digi**: XB24CZ7PIT-004	Módulos Zigbee / 802.15.4 XBee ZB S2C TH PCB Antena	14'59€	7	102'13€

Mouser: 803-PR103J2 **US Sensor:** PR103J2	Termistor NTC 10KOhms	3’53€	40	141’2€
Mouser: 279-YR1B210KCC **TE Connectivity:** YR1B210KCC	Resistencia 210KOhms 1/2W 0.1% Axial	0’29	40	11’6€
Mouser: 511-LD1117AV33 **STMicroelectronics:** LD1117AV33	Reguladores de voltaje LDO 3.3V 1.0A	0’61€	9	5’49€
Mouser: 647-ULD1C101MED1TD **Nichion:** ULD1C101MED1TD	Capacitadores electrolíticos de aluminio - Con patas 16V 100UF 20%	0’11€	9	0’99€
Mouser: 534-232 **Keystone:** 232	Conectores y contactos para pilas de 9 V 4" BATTERY SNAP	0’70€	9	6’3€
Mouser: 485-366 **Adafruit:** 366	2mm 10 pin Socket Header	0’78€	18	14’04
		Total, sin impuestos:		**281’75€**
		Impuestos (21%):		**51’17€**
		Gastos de importación (20%):		**56’35€**
Amazon: B01FS4KDN6	Tornillo de montaje Uctop Store de 2 polos, 2,54 mm, PCB, para conectar bloques de terminales, 30 unidades	11’3€ (IVA incl.)	**1**	**11’3€**
Amazon: B00149O6ZQ	VARTA Industrial 4022 - Pilas alcalinas 9 V/E-Block/6LR61, pack de 20 unidades	20’29€ (IVA incl.)	**1**	**20’29€**
Amazon: B07B8XSH65	Enisina. Interruptor de palanca con cable, 3A 250V AC/6A 125V AC MTS-1, 10pcs	11’99€ (IVA incl.)	**1**	**11’99€**
Amazon: B06X977K8L	52 piezas Neodimio Imán 10x2 mm. Extrem Fuerte 2,2 kg de fuerza. Magenesis	9’95€ (IVA incl.)	**1**	**9’95€**
Amazon: B013UDL5V6	Tarjeta de memoria microSDHC UHS-I de 16 GB con adaptador SD para raspberri pi model b	9’99€ (IVA incl.)	**1**	**9’99**
Amazon: B00T2U7R7I	Raspberry Pi 2 Model B - Placa base (ARM Quad-Core 900 MHz, 1 GB RAM, 4 x USB, HDMI, RJ-45) + caja + alimentador	49’25€ (IVA incl.)	**1**	**49’25**
JLCPB	Pack de 15 PCBs para fabricación de los nodos (más gastos de envío)	31’09€ (IVA incl.)	**1**	**31’09€**
Rs Pro: 832-0226	Filamento para impresora 3D RS Pro Azul 1.75mm, 1kg PLA	37’5€ (IVA incl.)	**2**	**75€**
	Material variado: Tornillos, pintura, estaño, cinta kapton.		**1**	**10€**
PRECIO TOTAL FABRICACIÓN DE LOS NODOS* **Nótese que sólo se compran 7 módulos XBee porque se utilizan los adquiridos en el Digi Mesh Kit de la fase de prototipado*				**618’13€**

8.3 Horas de Ingeniería

Para la estimación del coste debido al tiempo de dedicación se ha utilizado el salario bruto medio anual para un ingeniero de telecomunicaciones, y de allí, asumiendo un salario promedio de 25000€ bruto/año (se ha tomado como fuentes a jobtonic [96] e indeed [97]) y si consideramos que las horas efectivas según el estatuto de los trabajadores de 1780 horas (puede variar si existe convenio colectivo), obtenemos que el precio/hora es de 14'04€ brutos /hora.

Producto	PVPu	Cantidad	Total
Horas de ingeniería	14'04€ (impuestos incluidos)	300	4212€

8.4 Material de *backup*

Dado que los plazos de entrega de los productos adquiridos siempre son críticos para un proyecto como este, máxime cuando ciertos materiales deben ser importados, se decidió la compra de componentes redundantes en previsión de fallo o avería de alguno de ellos.
El material de *backup* adquirido es el siguiente:

Identificador	Producto	PVPu	Cantidad	Total
Mouser: 888-XB24CZ7PIT-004 **Digi**: XB24CZ7PIT-004	Módulos Zigbee / 802.15.4 XBee ZB S2C TH PCB Antena	14'59€	4	58'36€
Mouser: 803-PR103J2 **US Sensor:** PR103J2	Termistor NTC 10KOhms	3'53€	4	14'12€
Mouser: 279-YR1B210KCC **TE Connectivity:** YR1B210KCC	Resistencia 210KOhms 1/2W 0.1% Axial	0'29	8	2'32€
Mouser: 511-LD1117AV33 **STMicroelectronics:** LD1117AV33	Reguladores de voltaje LDO 3.3V 1.0A	0'61€	4	2'44€
Mouser: 647-ULD1C101MED1TD **Nichion:** ULD1C101MED1TD	Capacitadores electrolíticos de aluminio - Con patas 16V 100UF 20%	0'11€	4	0'44€
Mouser: 485-366 **Adafruit:** 366	2mm 10 pin Socket Header	0'78€	4	3'12
		Total, sin impuestos:		**80'8€**
		Impuestos (21%):		**16'97€**
		Gastos de importación (20%):		**16'16€**
PRECIO TOTAL MATERIAL DE REPUESTO				**113'93€**

8.5 Coste total

A continuación, se muestra el coste total de la elaboración de la WSN, incluyéndose por un lado los materiales, tanto la fase de prototipado como del diseño final y *backup*, y por otro lado, los gastos asociados al coste en horas de ingeniería:

Concepto	Coste económico
Material para el desarrollo del equipo prototipo	167'86€
Material para el desarrollo de los nodos finales	618'13€
Horas de ingeniería	4212€
Material de repuesto	113'93€
TOTAL	**5111'92€**

9 Análisis de los resultados

Tal y como se ha indicado, la determinación del *seeing* local debido a las variaciones de temperatura en la D080 será un proceso de recogida de datos que abarque más allá del plazo fijado para este TFM. No obstante, sí se ha podido validar el principal objetivo del presente proyecto, que no era otro que determinar la posibilidad de la utilización de las WSNs para cuantificar el *seeing* de la cúpula D080 en el Observatorio Astrofísico de Javalambre

En las pruebas del sistema en producción, esto es, con todos los nodos ubicados en la cúpula, se han estado tomando datos de temperatura de forma continua, se hicieron pruebas de día y de noche en periodos que abarcaban minutos, horas y hasta días enteros, obteniendo valores coherentes con otros sensores auxiliares que se utilizan en la operación normal del telescopio T080 (tres sensores en la propia estructura del telescopio, y un sensor fijo conectado a PLC [98]).
Por otra parte, durante la fase de test, en ocasiones, produjeron pérdidas de conexión con algún nodo, se simularon posibles fallos mediante el apagado de los mismos de forma aleatoria o se probó la no instalación de algún nodo a pesar de estar listado en los archivos de configuración software (véase los anexos apartado *12.2.14 list_of_nodes.ini*). Tras las depuraciones y correcciones necesarias se ha conseguido un sistema robusto con un comportamiento predecible según lo diseñado, esto nos hace tener plena confianza en la validez de los productos desarrollados, puesto que, a esta robustez, se suman la precisión obtenida en las medidas y la versatilidad que proporciona la capacidad de ir ubicando de forma instantánea los nodos en distintos lugares según convenga o incluso ir añadiendo más si en el futuro se estima necesario.
Indicar también, como se verá en las conclusiones, que el sistema no está exento de posibles puntos de mejora y se pretende seguir trabajando en él para hacerlo cada día mejor, aumentando la robustez, flexibilidad y precisión.
A continuación, se presentan unas capturas de la adquisición de datos que tuvo lugar la noche del 31 de mayo del 2018 al 1 de junio de 2018 (También se ha comentado de las adversas condiciones meteorológicas que estamos sufriendo que han impedido, de momento, la obtención de más datos). Como se apreciará a pesar de haber tomado datos de temperatura durante toda la noche, sólo se capturaron imágenes astrofísicas durante poco más de un par de horas, es por ello por lo que se

presenta una gráfica de toda la noche y luego otras centradas en el periodo dónde se puedo observar y por tanto se obtuvieron valores de FWHM.

Ilustración 49. Captura de toda la noche de observación realizada por el nodo REMOTO_4. Como se observa en la gráfica de puntos azules, sólo se obtuvieron imágenes astrofísicas durante un breve periodo de tiempo de la noche analizada. La grafica superior muestra la diferencia entre pares de sensores (s1-s2) y (s3-s4), las cuatro gráficas inferiores son las temperaturas mostradas por sensor de un mismo nodo, mientras que arriba a la derecha tenemos los valores instantáneos de temperatura por sensor y tensión de alimentación. Muy posiblemente el origen de las diferencias de temperaturas entre sensores en esta imagen se deba a que, en cada nodo, los pares de sensores están ubicados de forma que uno toque la chapa metálica y otro esté separado unos 4cm de esta.

Ilustración 50. Captura del periodo de observación en la que hay imágenes (puntos azules) también realizada por el nodo REMOTO_4

Ilustración 51. Captura del periodo de observación en la que hay imágenes (puntos azules) también realizada por el nodo REMOTO_7

Ilustración 52. Captura del periodo de observación en la que hay imágenes (puntos azules) también realizada por el nodo REMOTO_9

Tal y como se observa, en las imágenes hay variaciones de la FWHM a lo largo de la operación, no obstante, todavía es pronto para determinar cuanta contribución a este empeoramiento es debida a las variaciones térmicas, ya que existen muchos factores que afectan a esta degradación de la calidad óptica, será necesario contar con un amplio número de registros para poder realizar estudios con un adecuado respaldo estadístico que nos permita obtener un conocimiento preciso de esta contribución.

10 Conclusiones

En primer lugar, se nombran las conclusiones obtenidas sobre la metodología y el planteamiento seguido para la realización de este trabajo y en segundo, sobre los

productos en sí mismos abarcando también, posibles puntos de mejora de cara al futuro.

Si hablamos sobre el planteamiento y/o metodología seguida podemos concluir que:

- El hecho de tener una idea inicial bien definida de lo que se quería conseguir ha ayudado a ser efectivos y centrar los esfuerzos de una manera productiva sin distracciones debidas a una falta de concreción de los objetivos.
- Por otra parte, dichos objetivos, desde el punto de vista del autor, han sido bastante ambiciosos sobre todo dado el plazo de tiempo del que se disponía, esto ha hecho que la carga de trabajo también haya sido muy elevada.
- Los plazos fijados también han sido muy exigentes y ha habido que recortar días a la fase de despliegue, principalmente por problemas de fabricación de las cajas y mejoras en el software, como, por ejemplo, la gestión multi-hilo del procedimiento de adquisición para cada nodo.
  No obstante, este posible retraso ya fue contemplado inicialmente, y se estimó que se podría utilizar parte del tiempo dedicado a la fase 4, “Memoria TFM” para realizar el despliegue de los nodos en el observatorio como así ha sido.
- Aunque los resultados de funcionamiento de todo el sistema han sido plenamente satisfactorios, es necesario remarcar que será necesario disponer de suficientes datos estadísticos para determinar la relación que existe entre la degradación del seeing y las variaciones de temperatura. Esto implica que serán necesarios varios meses de adquisición de datos para determinar la contribución de la cúpula D080 al *seeing* local y la realización de un estudio posterior basándonos en una amplia estadística. No obstante, tal y como se ha dicho, esto queda fuera del alcance de este proyecto, en el que nos fijábamos como objetivo determinar si es posible medir la contribución al seeing local de la cúpula D080 con una WSN, obteniendo, desde el punto de vista del autor una respuesta totalmente favorable, ya que hemos sido capaces de medir durante plazos prolongados de tiempo 36 temperaturas diferentes y las diferencias entre pares de ellas, mediante 9 nodos ubicados de forma variable en dicha cúpula D080, lo que nos hace, basándonos en todos los estudios similares para la determinación del seeing de cúpula (de los que ya se ha hablado en esta memoria), tener confianza en la validez del sistema aquí desarrollado para la determinación del *seeing* local.

En cuanto a la realización de los productos en sí y a las decisiones tomadas en general se puede decir que, aunque se han logrado los objetivos propuestos, existen claros aspectos críticos y puntos de mejora:

- Durante las fases iniciales de prototipado se presentó un problema de inestabilidad en las medidas capturadas por los nodos, de forma que los valores calculados oscilaban continuamente en varios grados y se presentaban valores totalmente imprecisos. Se inició una incidencia con el soporte al cliente que Digi presta desde Estados Unidos al resto del mundo, y tras una serie de pruebas se concluyó que el problema no era de los módulos XBee sino debido al propio *setup* del sistema de pruebas (se estaba utilizando una *breadboard* para la realización de los test). En ese momento se decidió utilizar una placa pre-taladrada y soldar todos los componentes con un resultado totalmente satisfactorio.

- Otro problema presentado es que no se ha podido contrastar frente a un sistema más fiable las medidas de los nodos. Para comprobar si los valores eran correctos, dichas mediciones se comparaban contra una estación meteorológica y un termómetro por infrarrojos, no obstante, se confía más en el sistema diseñado al ser los productos de mayor calidad. Indicar también que, por una parte, se han medido las temperaturas de varios puntos con distintos nodos y se ha medido también la resistencia que presentan los termistores, y en todas las pruebas se han obtenido valores coherentes que nos hacen confiar en la calidad de las medidas obtenidas.

- Desde el punto de vista de este autor, el principal problema que presenta el presente proyecto es que el protocolo y los dispositivos XBee elegidos carecen de una gestión interna y sincronizada del modo sueño, esto es, en nuestra configuración cada nodo entra en modo sueño y no despierta hasta que su propio reloj interno le marca que debe hacerlo. Dado que dichos relojes trabajan de forma autónoma, no hay manera dentro del protocolo de sincronizar los equipos para tomar datos al unísono, esto es un problema para largos periodos de sueño dónde se le pueden pedir datos a un nodo y este no responder hasta que despierte, mientras que otro nodo puede estar despierto y responder con sus datos en dicho momento.
  Es cierto que estos módulos admiten la activación desde el modo sueño mediante la utilización de un pin externo, sin embargo, esto requeriría el uso de otro microcontrolador para realizar esta gestión, lo que se desestimó en las primeras fases del presente TFM ya que podría ir en contra de la búsqueda del máximo ahorro de energía.
  Se plantea aquí, como solución la utilización del protocolo Digi Mesh en lugar de Zigbee que permite la activación por RF en lo que se conoce como modo sueño síncrono y que además hace una gestión muy eficaz de la energía, sin embargo, los dispositivos adquiridos, aunque admiten este protocolo mediante una simple actualización del *firmware* no admiten este modo sueño sincronizado.
  En el presente proyecto se ha fijado el modo sueño a 5s, y se ha estado monitorizando el desfase entre medidas de distintos nodos sin superarse en la práctica desfases de más de 3s. No obstante, para cuando se requieren medidas más precisas se puede desactivar el modo sueño, eso sí, a costa del aumento en el consumo de las baterías.
- Otro punto que preocupa al autor es la autonomía de los nodos, dado que con la duración del modo sueño elegida (5s) el fabricante prevé una duración casi 55 días, aunque a esto hay que sumarle el consumo del resto de la electrónica. Se plantea de cara al futuro utilizar baterías de mayor capacidad pudiendo multiplicar x3 o x4 la duración de las actuales.
  También se concluye que para capturas con mayor frecuencia de muestreo el consumo de energía no es despreciable y para estos casos debería reconsiderarse si es un coste asumible (lo que implica cambios de baterías más frecuentemente) o por el contrario el sistema no es válido y por tanto se debería rehacer. Desde nuestro punto de vista, para frecuencias de muestreo mucho mayores sería razonable estudiar otro protocolo y otros sensores con mucho menor consumo.
- También existe un amplio margen de mejora en la realización de las cajas 3D para albergar la electrónica, sobre todo en lo referente a acabados y

estanqueidad. Se plantea rehacer el diseño para disminuir el grosor de las paredes y utilizar gomas de caucho en todas las juntas y orificios para permeabilizar los nodos lo máximo posible.

Finalmente indicar que, para este autor, ha sido muy sorprendente la poca aplicación práctica que las redes de sensores tienen en los observatorios astrofísicos profesionales, o al menos la falta de literatura al respecto, sin embargo, desde mi punto de vista, existe un amplísimo campo de aplicación en el que este tipo de redes serían de gran utilidad dentro de un centro como en el que desempeño mi actividad profesional.
Tras los conocimientos adquiridos en el desarrollo del presente proyecto se han abierto otros posibles campos de actuación dentro del propio observatorio sobre todo los relacionados con la monitorización de sistemas no críticos tales como temperaturas de salas, estados de suministros, activación y desactivación remota de sistemas y un largo etc.

11 Bibliografía

[1] colaboradores de Wikipedia, «Dispositivo de carga acoplada,» Wikipedia, La enciclopedia libre, 17 octubre 2017. [En línea]. Available: https://es.wikipedia.org/w/index.php?title=Dispositivo_de_carga_acoplada&oldid=102646023. [Último acceso: 11 marzo 2018].

[2] «Corrimiento al rojo,» colaboradores de Wikipedia, 28 febrero 2018. [En línea]. Available: https://es.wikipedia.org/w/index.php?title=Corrimiento_al_rojo&oldid=105894045. [Último acceso: 11 marzo 2018].

[3] W. contributors, «Atmospheric refraction,» Wikipedia, The Free Encyclopedia, 29 diciembre 2017. [En línea]. Available: https://en.wikipedia.org/w/index.php?title=Atmospheric_refraction&oldid=817536687. [Último acceso: 11 marzo 2018].

[4] B. MacEvoy, «Astronomical Seeing.Part 2: Seeing Measurement Methods,» Bruce MacEvoy, 11 noviembre 2013. [En línea]. Available: https://www.handprint.com/ASTRO/seeing2.html. [Último acceso: 11 marzo 2018].

[5] I. A. d. C. (IAC), «Site Quality,» Instituto Astrofísico de Canarias (IAC), [En línea]. Available: http://www.ing.iac.es/astronomy/observing/conditions/. [Último acceso: 10 marzo 2018].

[6] Sánchez, S. F.; Aceituno, J.; Thiele, U.; Pérez-Ramírez, D.; Alves, J., «The Night Sky at the Calar Alto Observatory,» *The Publications of the Astronomical Society of the Pacific,* vol. 119, pp. 1186-1200, 10/2007.

[7] E. S. Observatory, «Paranal Site Information,» European Southern Observatory, [En línea]. Available: https://www.eso.org/sci/facilities/paranal/astroclimate/site.html. [Último acceso: 11 marzo 2018].

[8] Wikipedia contributors, «Mauna Kea Observatories,» Wikipedia, The Free Encyclopedia, 22 enero 2018. [En línea]. Available: https://en.wikipedia.org/w/index.php?title=Mauna_Kea_Observatories&oldid=821695070. [Último acceso: 2018 marzo 11].

[9] Erik Meza, Antonio Pereyra, Bruno Sicardy, Germán Comina, José Ishitsuka, «MEDIDAS DE CALIDAD DE CIELO (SEEING) USANDO LA TÉCNICA DIMM EN EL OBSERVATORIO DE HUANCAYO,» *Revista Cientifica TECNIA,* 2017/03/13.

[10] J. &. M.-T. C. Vernin, «Measuring astronomical seeing: The DA/IAC DIMM,» *Astronomical Society of the Pacific,* vol. 107, nº 709, pp. 265-272, 1995.

[11] D. Blanco, Estimating local seeing at DCT facility, College of Optical Sciences, the University of Arizona: E. University Blvd, Tucson, AZ 85721, 2008/08/27.

[12] c. d. Wikipedia, «Función de dispersión de punto,» Wikipedia, La enciclopedia libre, [En línea]. Available: https://es.wikipedia.org/w/index.php?title=Funci%C3%B3n_de_dispersi%C3%B3n_de_punto&oldid=96804225. [Último acceso: 15 05 2018].

[13] W. contributors, «Full width at half maximum,» Wikipedia, The Free Encyclopedia, [En línea]. Available: Wikipedia, The Free Encyclopedia. [Último acceso: 15 05 2018].

[14] SEXtractor, «SEXTractor,» SEXTractor, [En línea]. Available: https://www.astromatic.net/software/sextractor. [Último acceso: 05 06 2018].

[15] Leonel Guitierrez, Esteban Luna, «Telescopios e instrumentación para la observación astronómica,» revista.unam.mx, [En línea]. Available: http://www.revista.unam.mx/vol.5/num4/art23/art23-1c.htm. [Último acceso: 5 6 2018].

[16] I. ThermoAnalytics, «ThermoAnalytics. Total Thermal Solutions,» ThermoAnalytics, Inc, [En línea]. Available: http://www.thermoanalytics.com. [Último acceso: 11 marzo 2018].

[17] A. Dyer, «Simple Astrophotography: Piggybacking,» Sky & Telescope Media, 30 julio 2006. [En línea]. Available: http://www.skyandtelescope.com/astronomy-resources/simple-astrophotography/. [Último acceso: 11 marzo 2018].

[18] L. ZAGO, «THE EFFECT OF THE LOCAL ATMOSPHERIC ENVIRONMENT ON ASTRONOMICAL OBSERVATIONS,» THÈSE No 1394 (1995) ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE, LAUSANNE, 1995.

[19] I. A. d. Canarias, «William Herschel Telescope,» Instituto Astrofísico de Canarias, [En línea]. Available: http://www.ing.iac.es/astronomy/telescopes/wht/. [Último acceso: 16 marzo 2018].

[20] CFH, «Canada-France-Hawaii Telescope,» CFH, [En línea]. Available: http://www.cfht.hawaii.edu. [Último acceso: 16 marzo 2018].

[21] Derrick Salmo, David Cowley, And jerry sovka, «MIRROR, DOME, AND NATURAL SEEING AT CFHT,» *The Astronomical Society ofthe Pacific.,* vol. 103, pp. 1020-1032, 1991.

[22] G. Observatory, «Gemini Observatory,» Gemini Observatory, [En línea]. Available: https://www.gemini.edu/. [Último acceso: 16 marzo 2018].

[23] S. C. S. f. t. G. 8. Enclosure, «R. Ford,» Gemini Project Office, Tucson, Arizona 85719, December 14, 1993.

[24] Siemens, «NX I-deas TMG Thermal Analysis,» Siemens, [En línea]. Available: https://www.plm.automation.siemens.com/ko_kr/Images/2503_tcm72-4448.pdf. [Último acceso: 18 05 2018].

[25] OAC, «Osservatorio Astronomico di Cagliari,» OAC, [En línea]. Available: http://www.oa-cagliari.inaf.it/. [Último acceso: 16 marzo 2018].

[26] F. Buffa, G.C. Cocco, I. Porceddu, M. Serrau, «Dome seeing and temperature forecasting: a feasibility study for the Galileo Telescope,» *New Astronomy Reviews,* nº 42, pp. 447-449, 1998.

[27] R. O. o. Greenwich, «Royal Observatory of Greenwich,» [En línea]. Available: https://www.rmg.co.uk/royal-observatory. [Último acceso: 18 abril 2018].

[28] M.T. Bridgeland, C.R. Jenkins, «Measurements of mirror seeing in the laboratory and at the telescope,» *New Astronomy Reviews,* nº 42, pp. 435-440, 1998.

[29] W. contributors, «Shack–Hartmann wavefront sensor,» Wikipedia, The Free Encyclopedia., [En línea]. Available:

https://en.wikipedia.org/w/index.php?title=Shack%E2%80%93Hartmann_wavefront_sensor&oldid=761195700. [Último acceso: 16 marzo 2018].

[30] UNAM, «Observatorio Astronómico Nacional San Pedro Mártir,» UNAM, [En línea]. Available: http://www.astrossp.unam.mx/oanspm/index.php. [Último acceso: marzo 2018 17].

[31] *Revista mexicana de astronomía y astrofísica,* vol. 31, pp. 91-98, January 2007.

[32] L. Observatory, «Lowell Observatory - DCT Telesope,» Discovery Channel, [En línea]. Available: https://lowell.edu/research/research-facilities/4-3-meter-dct/. [Último acceso: 17 marzo 2018].

[33] IAC, «IAC,» IAC, [En línea]. Available: www.iac.es/. [Último acceso: 18 abril 18].

[34] J. J. Fuensalida ; B. García-Lorenzo ; C. Hoegemann, «Correction of the dome seeing contribution from generalized-SCIDAR data using evenness properties with Fourier analysis,» *Monthly Notices of the Royal Astronomical Society ,* vol. 389, nº 2, pp. 731-740, 2008.

[35] LBTO, «Large Binocular Telescope,» LBTO, [En línea]. Available: http://www.lbto.org/. [Último acceso: 2018 marzo 18].

[36] E. Masciadri, J. Stoesz, S. Hageli y F. Lascaux, «Optical turbulence vertical distribution with standard and high resolution at Mt Graham,» *Monthly Notices of the ROYAL ASTRONOMICAL SOCIETY,* nº 404, pp. 144-158, 2010.

[37] M. University, «Sternberg Astronomical Institute,» Moscow University, [En línea]. Available: http://www.sai.msu.su/. [Último acceso: 18 marzo 2018].

[38] S. A. Potanin, «Shack–Hartmann Wavefront Sensor for Testing the Quality of the Optics of the 2.5-m SAI Telescope,» *Astronomy Reports,* vol. 86, nº 8, pp. 758-764, 2009.

[39] S. Potanin, «Estimation of the dome seeing from results of the optics quality tests with Shack-Hartman wavefront sensor,» *aSternberg Astronomical Institute Moscow State University,* 2011.

[40] W. contributors, «Dome F,» Wikipedia, The Free Encyclopedia, [En línea]. Available: https://en.wikipedia.org/w/index.php?title=Dome_F&oldid=799595640. [Último acceso: 18 marzo 2018].

[41] H. Okita, T. Ichikawa, M. C. B. Ashley, N. Takato, and H. Motoyama, «Excellent daytime seeing at Dome Fuji on the Antarctic plateau,» *Astronomy & AStrophysics,* nº 554, 2013.

[42] Andréss Guesalaga, Benoit Neichel, Angela Cortés, Clémentine Béchet and Dani Guzmán, «Using the C2n and wind profiler method with wide-field laser-guide-stars adaptive optics to quantify the frozen-flow decay,» *Monthly Notices of the ROYAL ASTRONOMICAL SOCIETY,* nº 440, pp. 1925-1933, 2014.

[43] W. contributors, «SOSUS,» Wikipedia, The Free Encyclopedia, 2 March 2018. [En línea]. Available: https://en.wikipedia.org/w/index.php?title=SOSUS&oldid=828397127. [Último acceso: 18 Abril 2018].

[44] Tongying Li y Zhenchao Zhang, «Data Collection Based on Mobile Agent in Wireless Sensor Networks,» Proceedings of the 10th World Congress on Intelligent Control and Automation, Beijing, China, July 6-8, 2012.

[45] Ian F. Akyildiz, Wilian Su, Yogesh Sankarasubramanian, Erdal Cayirci, «A survey on Sensor Networks,» *IEEE Communications Magazine,* vol. 40, nº 8, pp. 102-114, 2002.

[46] I. f. E. a. E. Engineers, «IEEE,» IEEE, [En línea]. Available: https://www.ieee.org/. [Último acceso: 02 05 2018].

[47] Kamal Deep SinghHakima ChaouchiJean Marie Bonnin, «Wireless sensor networks: a survey on recent developments and potential synergies,» *The Journal of Supercomputing,* vol. 68, pp. 1-48, April 2014.

[48] c. d. Wikipedia, « Espectro ensanchado,» Wikipedia, La enciclopedia libre, [En línea]. Available: https://es.wikipedia.org/w/index.php?title=Espectro_ensanchado&oldid=105246358. [Último acceso: 05 05 2018].

[49] W. contributors, « IEEE 802.15.4a,» Wikipedia, The Free Encyclopedia., [En línea]. Available: https://en.wikipedia.org/w/index.php?title=IEEE_802.15.4a&oldid=747122664. [Último acceso: 04 05 2018].

[50] Z. Alliance, «Zigbee,» Zigbee, [En línea]. Available: http://www.zigbee.org/. [Último acceso: 02 05 2018].

[51] c. d. Wikipedia, «Sistema embebido,» [En línea]. Available: https://es.wikipedia.org/w/index.php?title=Sistema_embebido&oldid=107283323. [Último acceso: 06 05 2018].

[52] I. E. T. Force, «IEFT,» IEFT, [En línea]. Available: https://www.ietf.org/. [Último acceso: 02 05 2018].

[53] W. contributors, «Highway Addressable Remote Transducer Protocol,» Wikipedia, The Free Encyclopedia, 29 diciembre 2017. [En línea]. Available: https://en.wikipedia.org/w/index.php?title=Highway_Addressable_Remote_Transducer_Protocol&oldid=817574713. [Último acceso: 14 marzo 2018].

[54] I. S. o. Automation, «ISA,» ISA, [En línea]. Available: https://www.isa.org/isa100/. [Último acceso: 02 05 2018].

[55] W. contributors, «Frequency-hopping spread spectrum,» Wikipedia, The Free Encyclopedia., [En línea]. Available: https://en.wikipedia.org/w/index.php?title=Frequency-hopping_spread_spectrum&oldid=839540031. [Último acceso: 05 05 2018].

[56] B. SIG, «Bluetooth SIG,» Bluetooth SIG Inc, [En línea]. Available: https://www.bluetooth.com/. [Último acceso: 03 05 2018].

[57] W. contributors, «Z-Wave,» Wikipedia, The Free Encyclopedia, [En línea]. Available: https://en.wikipedia.org/w/index.php?title=Z-Wave&oldid=838033888. [Último acceso: 06 05 2018].

[58] Z.-W. Alliance, «Z-Wave Allianze,» Z-Wave Allianze, 04 05 2018. [En línea]. Available: https://z-wavealliance.org/.

[59] A. Alliance, «ANT+ Alliance,» ANT+ Alliance, 04 05 2018. [En línea]. Available: https://www.thisisant.com/.

[60] Sabas, «medium.com,» medium.com, [En línea]. Available: https://medium.com/beelan/haciendo-iot-con-lora-cap%C3%ADtulo-1-qu%C3%A9-es-lora-y-lorawan-8c08d44208e8. [Último acceso: 06 05 2018].

[61] Honeywell, «Honeywell. Network Architecture,» Honeywell, [En línea]. Available: https://www.elstermetering.com/en/network-architecture. [Último acceso: 12 05 2018].

[62] SigFox, «SigFox,» SigFox, [En línea]. Available: https://www.sigfox.com/en. [Último acceso: 12 05 2018].

[63] D. Alliance, «DASH7 Alliance,» DASH7 Alliance, [En línea]. Available: http://www.dash7-alliance.org/.

[64] E. Alliance, «EnOcean Alliance,» EnOcean Alliance, EnOcean Alliance. [En línea]. Available: https://www.enocean-alliance.org/.

[65] c. d. Wikipedia, «RFID,» Wikipedia, La enciclopedia libre, 01 05 2018. [En línea]. Available: https://es.wikipedia.org/w/index.php?title=RFID&oldid=107470891. [Último acceso: 04 05 2018].

[66] Debmalya Bhattacharya and R.Krishnamoorthy, «Power Optimization in Wireless Sensor Networks,» *IJCSI International Journal of Computer Science Issues,,* vol. 8, nº 2, pp. 415-419, 2011.

[67] N. Semiconductors, «nRF5 SDK for Mesh,» Nordic Semiconductors, [En línea]. Available: https://www.nordicsemi.com/eng/Products/Bluetooth-low-energy/nRF5-SDK-for-Mesh. [Último acceso: 05 05 2018].

[68] D. Inc, «Introducing the XBee Zigbee Mesh Kit,» Digi Inc, [En línea]. Available: https://www.digi.com/blog/xbee/xbee-zigbee-mesh-kit/. [Último acceso: 05 05 2018].

[69] W. contributors, «Hirose U.FL,» Wikipedia, The Free Encyclopedia., [En línea]. Available: https://en.wikipedia.org/w/index.php?title=Hirose_U.FL&oldid=786980336. [Último acceso: 06 05 2018].

[70] c. d. Wikipedia, «SMA (conector),» Wikipedia, La enciclopedia libre., [En línea]. Available: https://es.wikipedia.org/w/index.php?title=SMA_(conector)&oldid=103292376. [Último acceso: 06 05 2018].

[71] c. d. Wikipedia, «Modulación por ancho de pulsos,» Wikipedia, La enciclopedia libre., [En línea]. Available: https://es.wikipedia.org/w/index.php?title=Modulaci%C3%B3n_por_ancho_de_pulsos&oldid=103557428. [Último acceso: 12 06 2018].

[72] D. Inc, «Digi Inc,» [En línea]. Available: https://www.digi.com/resources/documentation/digidocs/pdfs/90002002.pdf. [Último acceso: 01 01 2018].

[73] A. Devices, «TMP36,» [En línea]. Available: http://www.analog.com/media/en/technical-documentation/data-sheets/TMP35_36_37.pdf. [Último acceso: 05 05 2018].

[74] T. Instruments, «lmt84,» [En línea]. Available: http://www.ti.com/lit/ds/symlink/lmt84.pdf. [Último acceso: 05 05 2018].

[75] U. Sensor, «Thermistor products,» [En línea]. Available: http://www.ussensor.com/. [Último acceso: 05 05 2018].

[76] W. contributors, «Steinhart–Hart equation,» [En línea]. Available: https://en.wikipedia.org/w/index.php?title=Steinhart%E2%80%93Hart_equation&oldid=818958657. [Último acceso: 05 05 2018].

[77] JLCPCB, «JLCPCB,» [En línea]. Available: https://jlcpcb.com/. [Último acceso: 03 03 2018].

[78] S. Online, «Sensors Online,» Sensors Online, [En línea]. Available: https://www.sensorsmag.com/components/a-practical-guide-to-battery-technologies-for-wireless-sensor-networking. [Último acceso: 12 05 2018].

[79] Digi, «Digi Power Life Calculator,» [En línea]. Available: ftp://ftp1.digi.com/support/utilities/batterylifecalculator.xls. [Último acceso: 12 05 2018].

[80] UVA, «¿Qué es el ácido poliláctico?,» [En línea]. Available: http://www.eis.uva.es/~biopolimeros/alberto/pla.htm. [Último acceso: 13 05 2018].

[81] D. Inc, «XCTU,» [En línea]. Available: https://www.digi.com/products/xbee-rf-solutions/xctu-software/xctu. [Último acceso: 13 05 2018].

[82] c. d. Wikipedia, «Protocolos y arquitectura de X Window System,» Wikipedia, La enciclopedia libre., [En línea]. Available: https://es.wikipedia.org/w/index.php?title=Protocolos_y_arquitectura_de_X_Window_System&oldid=107361885. [Último acceso: 14 05 2018].

[83] Python.org, «Descarga Python,» Python.org, [En línea]. Available: https://www.python.org/downloads/. [Último acceso: 31 05 2018].

[84] Influxdata, «influxdata,» influxdata, [En línea]. Available: https://www.influxdata.com/. [Último acceso: 14 05 2018].

[85] G. Labs, «Grafana,» Grafana Labs, [En línea]. Available: https://grafana.com/. [Último acceso: 20 04 2018].

[86] Jetbrains, «Pycharm Python IDE,» Jetbrains, [En línea]. Available: https://www.jetbrains.com/pycharm/. [Último acceso: 13 05 2018].

[87] P. TM, «Tkinter,» Python TM, [En línea]. Available: https://wiki.python.org/moin/TkInter. [Último acceso: 14 05 2018].

[88] P. communicty, «PyPi,» Python community, [En línea]. Available: https://pypi.org/project/pip/#description. [Último acceso: 18 05 2018].

[89] Pasku, «github Pytelegraf,» github, [En línea]. Available: https://github.com/paksu/pytelegraf. [Último acceso: 10 04 2018].

[90] EasyEDA, «EasyEDA Designer,» EasyEDA, [En línea]. Available: https://easyeda.com/. [Último acceso: 12 03 2018].

[91] Autodesk, «AutoDesk TinkerCAD,» Autodesk, [En línea]. Available: https://www.tinkercad.com. [Último acceso: 15 05 2018].

[92] c. d. Wikipedia, «G-code,» Wikipedia, La enciclopedia libre, [En línea]. Available: https://es.wikipedia.org/w/index.php?title=G-code&oldid=106371032. [Último acceso: 18 05 2018].

[93] Fluke, «Multimetros Fluke Serie 89,» Fluke, 20 05 2018. [En línea]. Available: http://www.fluke.com/fluke/eses/multimetros-digitales/fluke-80-series-v-87v-83v-87v-e2-kit.htm?pid=56135.

[94] D.-K. electronics, «Digi-Key electronics,» Digi-Key electronics, [En línea]. Available: https://www.digikey.es. [Último acceso: 21 05 2018].

[95] M. Electronics, «Mouser Electronics,» Mouser Electronics, [En línea]. Available: https://www.mouser.es/. [Último acceso: 16 05 2018].

[96] Jobtonic, «Jobtonic,» Jobtonic, [En línea]. Available: http://espana.jobtonic.es/salary/26526/16921.html. [Último acceso: 22 05 2018].

[97] Indeed, «Indeed,» Indeed, [En línea]. Available: https://www.indeed.es/salaries/Ingeniero/a-en-telecomunicaciones-Salaries. [Último acceso: 22 5 2018].

[98] VAISALA, «Vaisala HMP155,» Vaisala, [En línea]. Available: https://www.vaisala.com/sites/default/files/documents/HMP155-Datasheet-B210752EN-E-LoRes.pdf. [Último acceso: 01 06 2018].

[99] S. Rueda, «Sergio Rueda TFM 2018. UOC,» github, [En línea]. Available: https://github.com/sruedat/TFM_Srueda.git. [Último acceso: 10 06 2018].

[100] Waltenegus Dargie, Christian Poellabauer, Fundamentals of Wireless Sensor Networks Theory and Practice, John Wiley & Sons Ltd, 2010.

[101] W. contributors, «Digital signal processor,» Wikipedia, The Free Encyclopedia, 9 marzo 2018. [En línea]. Available: https://en.wikipedia.org/w/index.php?title=Digital_signal_processor&oldid=829522878. [Último acceso: 13 marzo 2018].

[102] W. contributors, «Field-programmable gate array,» Wikipedia, The Free Encyclopedia, 9 marzo 2018. [En línea]. Available: https://en.wikipedia.org/w/index.php?title=Field-programmable_gate_array&oldid=829623276. [Último acceso: 13 marzo 2018].

[103] R. Hawi, «Wireless Sensor Networks – Sensor Node Architecture and Design Challenges,» *International Journal of Advanced Research in Computer Science,* vol. 5, nº 1, p. 7, 2014.

[104] Antônio Dâmaso, Davi Freitas, Nelson Rosa , Bruno Silva and Paulo Maciel, «Evaluating the Power Consumption of Wireless Sensor Network Applications Using Models,» *Sensors,* p. 9, 2013.

[105] Wenqi GUO and William M. HEALY, «Power Supply Issues in Battery Reliant Wireless Sensor Networks: A Review,» *INTERNATIONAL JOURNAL OF INTELLIGENT CONTROL AND SYSTEMS,* vol. 19, nº 1, pp. 15-23, March 2014.

[106] Sandeep Mehra, CN Khairna, «ANALYSIS AND DESIGN OF ULTRA LOW POWER ADC FOR WIRELESS SENSOR NETWORKS,» *INTERNATIONAL JOURNAL OF ELECTRONICS AND COMMUNICATION ENGINEERING & TECHNOLOGY (IJECET),* vol. 4, nº 1, pp. 264-275, 2013.

[107] P. M. Pathak, «An Approach to Memory management in Wireless Sensor Networks,» *International Journal of Computer Science & Engineering Technology (IJCSET),* vol. 4, pp. 1171-1176, 08 Aug 2013.

[108] D. Schroeder, «Adaptive low-power analog/digital converters for wireless sensor networks,» *Third International Workshop on Intelligent Solutions in Embedded Systems,* pp. 70-78, 2005.

[109] W. contributors, «Sensor node,» Wikipedia, The Free Encyclopedia, 25 octubre 2016. [En línea]. Available: https://en.wikipedia.org/w/index.php?title=Sensor_node&oldid=746158710. [Último acceso: 13 marzo 2018].

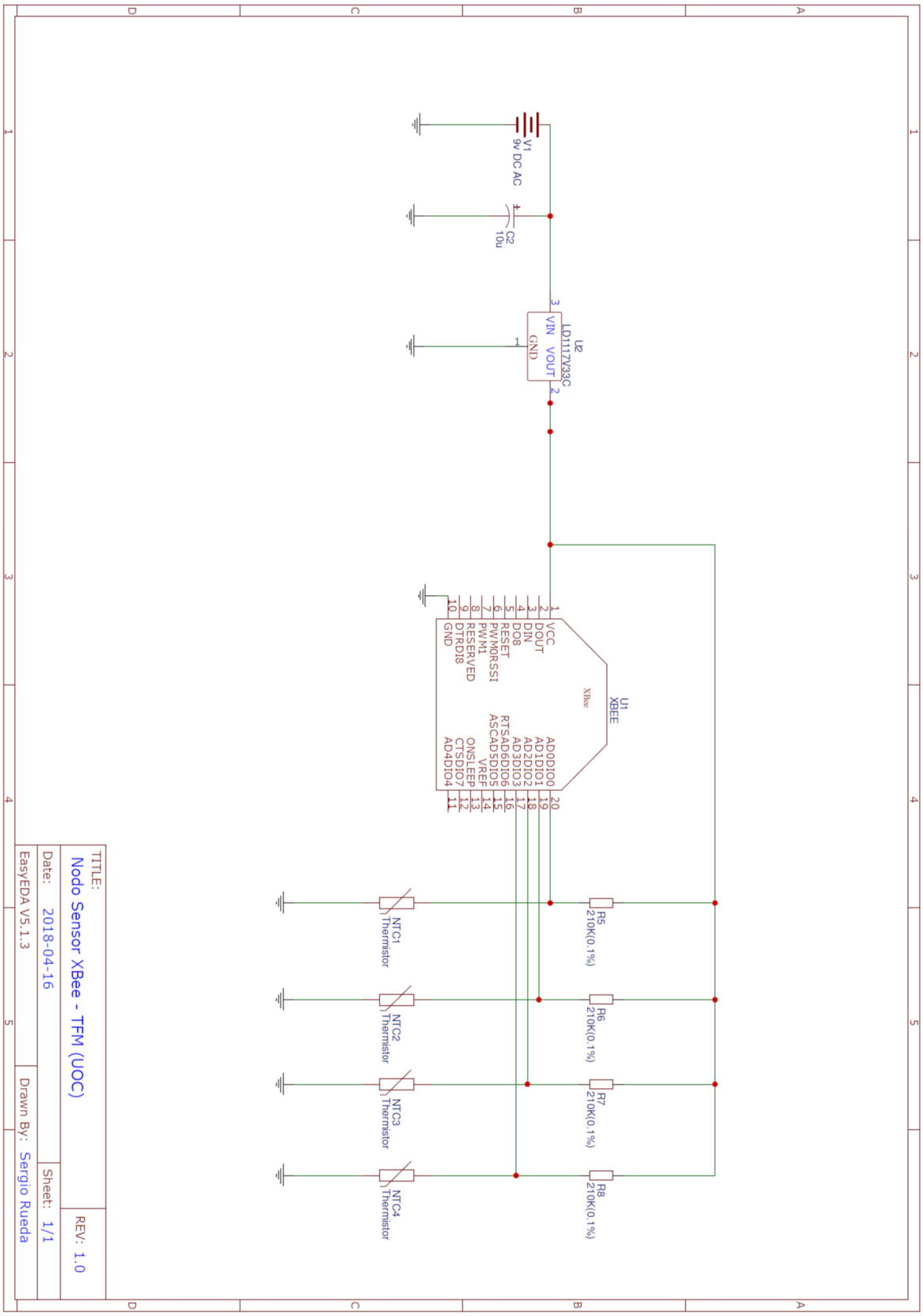
[110] Divya Sharma, Sandeep Verma, Kanika Sharma, «Network Topologies in Wireless Sensor Networks: A Review,» *International Journal of Electronics & Communication Technology (IJECT),* vol. 4, nº 3, abril 2013.

[111] D. Inc, «Introducing The Official XBee Java Library,» Digi Inc, [En línea]. Available: https://www.digi.com/blog/community/official-xbee-java-library/. [Último acceso: 13 05 2018].

[112] D. Inc, «Introducing the Official Digi XBee Python Library,» Digi Inc, [En línea]. Available: https://www.digi.com/blog/xbee/introducing-the-official-digi-xbee-python-library/. [Último acceso: 13 05 2018].

12 Anexos

12.1 Esquema eléctrico



12.2 Módulos Python3 desarrollados

A continuación, se adjuntan todos los códigos elaborados para la realización de este TFM, si bien dichos códigos están disponibles para su descarga en *github* [99].

12.2.1 discover_devices_xbee.py

```
# Sergio Rueda Teruel. 2018
# Este software ha sido desarrollado para el trabajo fin de master de la titulación
# Máster Universitario en Ingeniería de Telecomunicación UOC-URL de la
# Universidad Oberta de Catalunya y lleva por título
# "Diseño de una WSN para la estimación del seeing de la cúpula D080,
# en el Observatorio Astrofísico de Javalambre."

# Para la realización de este código se han utilizado las librerías Python
# que la empresa Digi (Digi International Inc.) proporciona en su página web
# (https://www.digi.com/blog/xbee/introducing-the-official-digi-xbee-python-library/)

# este código está sometido a licencia de Reconocimiento-NoComercial-CompartirIgual
# 3.0 España de Creative Commons.



# Este módulo se utiliza para descubrir equipos remotos conectados a la misma red (PAN ID)
# que el equipo local, es desde este equipo local desde el que se realizan las tareas de
# descubrimiento.

import time, sys, argparse
import read_sys_config
from digi.xbee.models.status import NetworkDiscoveryStatus
from digi.xbee.devices import XBeeDevice
sys.tracebacklimit = 0

# Se admiten parámetros de depuración para la invocacción directa del módulo
# el parámetro -v VERBOSE muestra por consola la evolución del módulo
# discover-devices_xbee -v VERBOSE
parser= argparse.ArgumentParser()
parser.add_argument("-v","--verbose", help="Activates Verbose output")
args=parser.parse_args()

# Parámetros de conexión con el puerto serie al dispositivo local
port = read_sys_config.ReadLocalPortFromFile()
baud_rate = read_sys_config.ReadLocalBaudRateFromFile()



# La variable log será utilizada para ir registrando el avance de todo el proceso
# y se devolverá en el procedimiento passlog para que pueda ser leída accedida
# por otros módulos
log=""

# Este procedimiento puede ser invocado por otros módulos para ir conociendo el estado
# del proceso de descubrimiento de nodos en la red
def passlog():
    global log
    time.sleep(0.2)
    return log

# Procedimiento principal que gestiona las tareas de descubrimiento
def main():
    global log
    state = 0
    if args.verbose:
        print(" +---------------------------------------------+")
        print(" |       XBee Discover Devices Module          |")
        print(" +---------------------------------------------+\n")

    local_device = XBeeDevice(port, baud_rate)
    try:
        log=""
        local_device.open()

        xbee_network = local_device.get_network()
```

```
        xbee_network.set_discovery_timeout(15)  # 15 segundos

        xbee_network.clear()

        # Callback para los dispositivos descubiertos
        def callback_device_discovered(remote):
            global log
            log =log+"Device discovered:\n" + "%s" % remote +"\n\n"
            if args.verbose:
                print("Device discovered: " + "%s" % remote)

        # Callback para la finalización del proceso de descubrimiento
        def callback_discovery_finished(status):
            global log
            if status == NetworkDiscoveryStatus.SUCCESS:
                log=log+"Discovery process finished successfully.\n"

                if args.verbose:
                    print("Discovery process finished successfully.")
            else:
                log= "There was an error discovering devices: %s\n" % status.description
                if args.verbose:
                    print("There was an error discovering devices: %s" % status.description)

        xbee_network.add_device_discovered_callback(callback_device_discovered)

        xbee_network.add_discovery_process_finished_callback(callback_discovery_finished)

        xbee_network.start_discovery_process()

        log = "\nDiscovering remote XBee devices...\n\n"
        if args.verbose:
            print("Discovering remote XBee devices...")


        while xbee_network.is_discovery_running():
            # Actualiza el log para otros módulos
            passlog()
            time.sleep(0.1)

        # Si es invocado por otros módulos devuelve un 1 indicando que ya ha acabado la fase
de descubrimiento
        state = 1
        return state


    except:
        if local_device.is_open():
            local_device.close()
        pass



    finally:
        if local_device is not None and local_device.is_open():
            local_device.close()
        sys._clear_type_cache()



if __name__ == '__main__':
    main()
```

12.2.2 get_fwhm.py

```
# Sergio Rueda Teruel. 2018
# Este software ha sido desarrollado para el trabajo fin de master de la titulación
# Máster Universitario en Ingeniería de Telecomunicación UOC-URL de la
# Universidad Oberta de Catalunya y lleva por título
# "Diseño de una WSN para la estimación del seeing de la cúpula D080,
# en el Observatorio Astrofísico de Javalambre."
```

```
# Para la realización de este código se han utilizado las recomendaciones y utilidades
# que los ingenieros de bases de datos del OAJ han desarrollado.
# Agradecer la ayuda de Javier Hernández su ayuda en la elaboración de módulo

# Este código está sometido a licencia de Reconocimiento-NoComercial-CompartirIgual
# 3.0 España de Creative Commons.

# Este módulo se utiliza para consultar a la base de datos del centro de cálculo sobre
# el valor de FWHM de las imágenes capturadas, esta consulta se realiza en función de
# la fecha introducida en la variable FECHA, telegráf introducirá los datos con el
# timestamp que viene en la propia imagen obtenida con lo que no importa cuando se ejecute
# este módulo ya que siempre insertará el valor de FWHM con la fecha correcta


import math
import json
import urllib.request
import datetime
import time
import send_data_to_telegraf

#Editamos el valor de la fecha dependiendo desde cuando queremos obtener la imágenes
FECHA = datetime.datetime(2018, 5, 31, 16, 0)

FILTER_LAMBDA = {'J0430': 430, 'gSDSS': 475, 'rSDSS': 625, 'iSDSS': 773, 'J0861': 861,
'zSDSS': 915, 'uJAVA': 348,
                 'J0378': 378, 'J0395': 395, 'J0410': 410, 'J0515': 515, 'J0660': 660}
FILTER_ZP = {'uJAVA': 21.80, 'J0378': 21.00, 'J0395': 20.80, 'J0410': 21.64, 'J0430': 21.61,
'gSDSS': 23.70,
             'J0515': 21.65, 'rSDSS': 23.70, 'J0660': 21.10, 'iSDSS': 23.40, 'J0861': 21.60,
'zSDSS': 22.60}

SQL_IMAGES = """SELECT oa.name, TIMESTAMP(oa.date, oa.time) AS datetime, fwhmg, fwhm_mean,
f.name, airmass, INSERTDATE_CR
FROM t80oa oa JOIN rc ON oa.id = rc.ori_id join filter f ON oa.filter_id = f.id WHERE FWHMG IS
NOT NULL
AND oa.date >= DATE('{0}') AND oa.time > TIME('{1}') ORDER BY 2"""

SQL_SEEING = """SELECT datetime, seeing FROM seeing.seeingdata WHERE datetime >=
TIMESTAMP(DATE('{0}'), TIME('{1}')) AND flag = 0 ORDER BY 1"""



def query_adql_by_http(query, url):
    data = json.dumps({'query': query})
    content = urllib.request.urlopen(url, bytearray(data, 'utf-8')).read()
    d = json.loads(content.decode('utf-8'))
    if d.get('error'):
        raise Exception(d['error'])

    return d['rows']


def get_seeing_data(timestamp, url):
    """
    Query for the available 'seeing' data form five minutes before the 'timestamp'.
    """
    t2 = (timestamp - datetime.timedelta(minutes=5)).isoformat()
    sql = SQL_SEEING.format(t2[:10], t2[11:16])
    res = query_adql_by_http(sql, url)

    return [(datetime.datetime.strptime(x[0], '%Y-%m-%dT%H:%M:%S'), float(x[1])) for x in res]


def get_seeing_at(timestamp, seeing):
    """Get the median seeing from measures 5 minutes before and after the 'timestamp'"""
    t_min = timestamp - datetime.timedelta(minutes=5)
    t_max = timestamp + datetime.timedelta(minutes=5)
    s_data = []
    for s in seeing:
        if s[0] >= t_min and s[0] <= t_max:
            s_data.append(s[1])

    if len(s_data) == 0:
        return None

    s_data.sort()
```

```
    return s_data[int(len(s_data) / 2)]


def get_expected_psf(img, seeing):
    s = get_seeing_at(img['timestamp'], seeing)
    if s is None: return None

    return s * math.pow(img['airmass'], 0.6) * (math.pow(FILTER_LAMBDA[img['filter']], -0.2) /
0.2885399811814427)


def query_images_since(timestamp,
url='http://upad.oaj.cefca.es/reduction/t80commcpd/exec_adql'):
    result = []
    seeing = get_seeing_data(timestamp, url)
    itmtp = timestamp.isoformat()
    sql = SQL_IMAGES.format(itmtp[:10], itmtp[11:16])
    res = query_adql_by_http(sql, url)
    for i in range(len(res)):
        cur = res[i]
        img = {'name': cur[0],
               'timestamp': datetime.datetime.strptime(cur[1], '%Y-%m-%dT%H:%M:%S'),
               'fwhmg': float(cur[2]), 'fwhm_mean': cur[3],
               'filter': cur[4], 'airmass': float(cur[5]),
               'insertdate': datetime.datetime.strptime(cur[6], '%Y-%m-%dT%H:%M:%S')}


        img['expected_psf'] = get_expected_psf(img, seeing)

        result.append(img)

    return result


if __name__ == '__main__':
    last_img=[]
    result = query_images_since(FECHA)
    print('Total %d' % len(result))
    if len(result) > 0:
        for i in range (len(result)):
            img = result[i]
            print('Image %s' % i)
            print('   name: %s' % img['name'])
            print('   filter %s' % img['filter'])
            print('   timestamp: %s' % img['timestamp'])
            print(img['timestamp'])
            print('   fwhmg (PSF): %s' % img['fwhmg'])
            print('   expected_psf: %s' % img['expected_psf'])

send_data_to_telegraf.send_fwhm_with_timestamp(float(img['fwhmg']),time.mktime(img['timestamp'
].timetuple()))
            time.sleep(0.1)

    else:
        print('No new imagenes available')
```

12.2.3 gui.py

```
# Sergio Rueda Teruel. 2018
# Este software ha sido desarrollado para el trabajo fin de máster de la titulación
# Máster Universitario en Ingeniería de Telecomunicación UOC-URL de la
# Universidad Oberta de Catalunya y lleva por título
# "Diseño de una WSN para la estimación del seeing de la cúpula D080,
# en el Observatorio Astrofísico de Javalambre."
# Este código está sometido a licencia de Reconocimiento-NoComercial-CompartirIgual
# 3.0 España de Creative Commons.

# Este módulo se utiliza para la creación de la interface gráfica con la que gestionar
# toda la red, permite leer la configuración del nodo local, la configuración de nodos
# remotos indicando su dirección. Además, también permite escribir la configuración del
# nodo local y/o remotos mediante sus correspondientes archivos de configuración
```

```
import tkinter as tk
from tkinter import ttk
import threading
import queue
import time
import os
import platform
import subprocess

# módulos espcificamente desarrollados para este TFM
import discover_devices_xbee    # Este módulo gestiona el decubrimiento de nodos remotos vía
RF
import read_local_params_xbee   # Desde este módulo se leen los parámetros del nodo conectado
al puerto serie
import read_remote_params_xbee  # Desde este módulo se leen los parámetros de un nodo remoto
vía RF
import write_remote_params_xbee # Con este módulo se escriben parámetros a un nodo remoto vía
RF
import write_local_params_xbee  # Con este módulo se escriben parámetros al nodo local por
puerto serie


# Aplicación
class Application(tk.Frame):
    # Creación del master frame que alberga todos los componentes
    def __init__(self, master=None):
        tk.Frame.__init__(self, master)
        master.title("SRueda. - Xbee Zigbee Tool")
        master.columnconfigure(1, weight=1)
        master.columnconfigure(1, weight=1)
        master.rowconfigure(1, weight=1)
        master.rowconfigure(1, weight=1)
        self.queue = queue.Queue()
        self.create_widgets()

    # Función para definir el directorio dónde están los arhivos de configuración que son
leídos por varios de los
    # módulos importados, es válido para SO Windows, Mac y GNU-Linux
    def open_config_file(self):
        dirname= os.path.dirname(__file__)+ "/config"
        if platform.system() == "Windows":
            os.startfile(dirname)
        elif platform.system() == "Darwin":
            subprocess.Popen(["open", dirname])
        else:
            subprocess.Popen(["xdg-open", dirname])

    # Desde esta función se habilitan todos los botones de la apliación ::::::::::::::::::
    def enable_all_bottons(self):
        self.discover['state'] = "normal"
        self.clear_discover['state'] = "normal"
        self.read_local_conf['state'] = "normal"
        self.write_local_conf['state'] = "normal"
        self.read_remote_conf['state'] = "normal"
        self.write_remote_conf['state'] = "normal"
        self.clear_params['state'] = "normal"
        self.open_file['state'] = "normal"
    #::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

    # Función para deshabilitar todos los botones de la aplicación ::::::::::::::::::::
    def disable_all_bottons(self):
        self.discover['state']=tk.DISABLED
        self.clear_discover['state']=tk.DISABLED
        self.read_local_conf['state']=tk.DISABLED
        self.write_local_conf['state']=tk.DISABLED
        self.read_remote_conf['state']=tk.DISABLED
        self.write_remote_conf['state']=tk.DISABLED
        self.clear_params['state']=tk.DISABLED
        self.open_file['state']=tk.DISABLED
    #::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

    # Desde estas funciones se realiza la lectura de parámetros de un módulo remoto,
    # la selección del módulo se realiza en base a su dirección MAC introducida en la GUI
    @staticmethod
    def read_remote_xbee_cmmd(self):
        # Comprobamos que la dirección del nodo introducida tenga una longitud
        # de 16 caracteres, si no es así, se completa con ceros a la izquierda
```

```
        data = self.remote_address.get()[:16]
        if len(data)<16:
            for n in range (len(data),16):
                data= "0" + data
        # Indicamos en el panel de texto que se va a intentar acceder al nodo remoto
        self.text_params.config(state="normal")
        self.text_params.insert(tk.INSERT,'Trying to access to remote device: ' + data +
'\n\n')
        # Se llama al módulo para la lectura de parámetros del nodo remoto pasándole la
dirección
        self.text_params.insert(tk.INSERT, read_remote_params_xbee.main(data))
        self.text_params.config(state=tk.DISABLED)
        self.enable_all_bottons()

    def read_remote_xbee(self):
        # Indicamos en el panel de texto que se va a lanzar la lectura del nodo remoto
        self.text_params.config(state="normal")
        self.text_params.delete('1.0', tk.END)
        self.text_params.insert(tk.INSERT, " \nReading remote node... \n\n")
        self.text_params.config(state=tk.DISABLED)
        self.disable_all_bottons()
        # Para evitar que la aplicación se bloque mientras se realiza la lectura del módulo
remoto
        # se hace la llamada en multihilo, así se realiza por un lado el mainloop y por el
otro la
        # lectura del nodo remoto
        threading.Thread(target=self.read_remote_xbee_cmmd,
                         args=(self, )).start()
    #:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

    # Desde estas funciones se realiza la escritura de parámetros a un módulo remoto,
    # la selección del módulo se realiza en base a su dirección MAC introducida en la GUI
    @staticmethod
    def write_remote_xbee_cmmd(self, boton1, boton2, boton3):
        self.text_params.config(state="normal")
        # the propper length of the address is 2 bytes, we must
        address = self.remote_address.get()[:16]
        if len(address)<16:
            for n in range (len(address),16):
                address= "0" + address
        self.text_params.insert(tk.INSERT,'Trying to access to remote device: ' + address +
'\n\n')
        self.text_params.insert(tk.INSERT, write_remote_params_xbee.main(address))
        self.text_params.config(state=tk.DISABLED)
        self.enable_all_bottons()

    def write_remote_xbee(self):
        self.text_params.config(state="normal")
        self.text_params.delete('1.0', tk.END)
        self.text_params.insert(tk.INSERT, " \nWriting remote node... \n\n")
        self.text_params.config(state=tk.DISABLED)
        self.disable_all_bottons()
        threading.Thread(target=self.write_remote_xbee_cmmd,
                         args=(self, self.read_local_conf, self.read_remote_conf,
self.clear_params,)).start()
    #:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::


    # Desde estas funciones se realiza la lectura de parámetros del módulo local conectado al
puerto serie
    def read_local_xbee(self):
        self.text_params.config(state="normal")
        self.text_params.delete('1.0', tk.END)
        self.text_params.insert(tk.INSERT, " \nReading local node... \n\n")
        self.text_params.config(state=tk.DISABLED)
        self.disable_all_bottons()
        threading.Thread(target=self.read_local_xbee_cmmd, args=(self,)).start()

    @staticmethod
    def read_local_xbee_cmmd(self):
        self.text_params.config(state="normal")
        self.text_params.insert(tk.INSERT, read_local_params_xbee.main())
        self.text_params.config(state=tk.DISABLED)
        self.enable_all_bottons()
    #:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

    # Desde estas funciones se realiza la escritura de parámetros del módulo local conectado
```

```
al puerto serie
    def write_local_xbee(self):
        self.text_params.config(state="normal")
        self.text_params.delete('1.0', tk.END)
        self.text_params.insert(tk.INSERT, " \nWriting local node... \n\n")
        self.text_params.config(state=tk.DISABLED)
        self.disable_all_bottons()
        threading.Thread(target=self.write_local_xbee_cmmd,
                         args=(self, )).start()

    @staticmethod
    def write_local_xbee_cmmd(self):
        self.text_params.config(state="normal")
        # the propper length of the address is 2 bytes, we must
        address = "local_node"
        self.text_params.insert(tk.INSERT, 'Trying to access to local device: ' + address +
'\n\n')
        self.text_params.insert(tk.INSERT, write_local_params_xbee.main(address))
        self.text_params.config(state=tk.DISABLED)
        self.enable_all_bottons()
    #:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

    #Esta función borra la información de los parámetros del nodo
    def clear_readed_params(self):
        self.text_params.config(state="normal")
        self.text_params.delete('1.0', tk.END)
        self.text_params.insert(tk.INSERT, '\n Please press "Read local conf." to read local
node params\n'
                                           ' or put address and press "Read remote conf." to
read remote node params')
        self.text_params.config(state=tk.DISABLED)
    #:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

    #Esta función borra la lista de nodos descubiertos
    def clear_discovered_nodes(self):
        self.text_discover.config(state="normal")
        self.text_discover.delete('1.0', tk.END)
        self.text_discover.insert(tk.INSERT, '\nPress "Search Nodes" to find remote XBees \n')
        self.text_discover.config(state=tk.DISABLED)
    #:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

    # Estas funciones actualizan los mensajes por pantalla en el proceso de descubrimiento de
los nodos remotos
    @staticmethod
    # Mientras que no se haya acabado el proceso de descubrimiento va leyendo el log y lo mete
en la cola
    def update_discover_log(self,cola):
        self.disable_all_bottons()
        while self.discover_status != 1:
            log=discover_devices_xbee.passlog()
            cola.put(log)
            time.sleep(0.1)

    # Si el status devuelto por el modulo para descubrir nodos es = 1 se acabó el proceso de
descubrimiento
    def search(self):
        self.discover_status=discover_devices_xbee.main()
        self.enable_all_bottons()

    # Gestión de las colas del log, mientras la cola no está vacía va sacando mensajes
    def process_queue(self):
        try:
            data = self.queue.get_nowait()
            self.text_discover.config(state="normal")
            self.text_discover.delete('1.0', tk.END)
            self.text_discover.insert(tk.INSERT,data)
        except queue.Empty:
            pass
        self.master.after(100, self.process_queue)

    # Función principal de la gestión de mensajes por pantalla en el proceso de
descubrimiento, se trata de unas
    # llamadas multi-hilo para la llamada al módulo importado desde el que se realiza el
proceso de descubrimiento
    # y para el llenado de la cola de mensajes del procedimiento
    def update_status(self):
        self.discover_status = 0
```

```
        self.text_discover.delete('1.0', tk.END)
        self.clear_discover['state']=tk.DISABLED
        threading.Thread(target=self.update_discover_log,
                         args=(self, self.queue,)).start()
        threading.Thread(target=self.search).start()
        self.after(100, self.process_queue)
    #::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::



    # Función para la creación del inteface gráfico en sí mismo, crea paneles, botones, etc.
    def create_widgets(self):
        # Panel izquierdo
        left_pane = tk.PanedWindow(root, orient=tk.VERTICAL)
        left_pane.grid(column=0, row=0, rowspan=2, sticky=(tk.N, tk.W, tk.E, tk.S))
        left_upperframe = ttk.Frame(root, relief='groove', borderwidth=2)
        left_upperframe.grid(column=0, row=0, sticky=(tk.N, tk.W, tk.E, tk.S))
        left_upperframe.columnconfigure(0, weight=1)
        left_upperframe.rowconfigure(0, weight=1)
        left_pane.add(left_upperframe, heigh=500, width=400)
        left_bottomframe = ttk.Frame(left_pane, relief='groove', borderwidth=2)
        left_bottomframe.grid(column=0, row=1, sticky=(tk.N, tk.W, tk.E, tk.S))
        left_bottomframe.columnconfigure(0, weight=1)
        left_bottomframe.rowconfigure(0, weight=1)
        left_pane.add(left_bottomframe, heigh=40)

        # Panel derecho
        right_pane = tk.PanedWindow(root, orient=tk.VERTICAL)
        right_pane.grid(column=1, row=0, rowspan=2, sticky=(tk.N, tk.W, tk.E, tk.S))
        right_upperframe = ttk.Frame(root, relief='groove', borderwidth=2)
        right_upperframe.grid(column=0, row=0, sticky=(tk.N, tk.W, tk.E, tk.S))
        right_upperframe.rowconfigure(0, weight=3)
        right_pane.add(right_upperframe, width=600, heigh=500)
        right_bottomframe = ttk.Frame(right_pane, relief='groove', borderwidth=2)
        right_bottomframe.grid(column=0, row=1, sticky=(tk.N, tk.W, tk.E, tk.S))
        right_bottomframe.columnconfigure(0, weight=1)
        right_bottomframe.rowconfigure(0, weight=1)
        right_pane.add(right_bottomframe)

        # Botones
        #discover
        self.discover = tk.Button(left_bottomframe)
        self.discover["text"] = "Search Nodes"
        self.discover["command"] = self.update_status
        self.discover.grid(column=0, row=0, sticky=(tk.W, tk.S, tk.E, tk.N))
        #clear discover
        self.clear_discover = tk.Button(left_bottomframe)
        self.clear_discover["text"] = "Clear"
        self.clear_discover["command"] = self.clear_discovered_nodes  #
self.UpdateDiscoveringState
        self.clear_discover.grid(column=1, row=0, sticky=(tk.W, tk.S, tk.E, tk.N))
        #read local configuration
        self.read_local_conf = tk.Button(right_bottomframe)
        self.read_local_conf["text"] = "Read local conf."
        self.read_local_conf["command"] = self.read_local_xbee
        self.read_local_conf.grid(column=2, row=1, sticky=(tk.W, tk.S, tk.E, tk.N))
        # write local configuration
        self.write_local_conf = tk.Button(right_bottomframe)
        self.write_local_conf["text"] = "Write local conf."
        self.write_local_conf["command"] = self.write_local_xbee
        self.write_local_conf.grid(column=3, row=1, sticky=(tk.W, tk.S, tk.E, tk.N))
        # read remote configuration
        self.read_remote_conf = tk.Button(right_bottomframe)
        self.read_remote_conf["text"] = "Read remote conf."
        self.read_remote_conf["command"] = self.read_remote_xbee
        self.read_remote_conf.grid(column=2, row=0, sticky=(tk.W, tk.S, tk.E, tk.N))
        # write remote configuration
        self.write_remote_conf = tk.Button(right_bottomframe)
        self.write_remote_conf["text"] = "Write remote conf."
        self.write_remote_conf["command"] = self.write_remote_xbee
        self.write_remote_conf.grid(column=3, row=0, sticky=(tk.W, tk.S, tk.E, tk.N))
        # clear params
        self.clear_params = tk.Button(right_bottomframe)
        self.clear_params["text"] = "Clear"
        self.clear_params["command"] = self.clear_readed_params
        self.clear_params.grid(column=4, row=0, sticky=(tk.W, tk.S, tk.E, tk.N))
```

```
        # open file
        self.open_file = tk.Button(right_bottomframe)
        self.open_file["text"] = "Open config files"
        self.open_file["command"] = self.open_config_file
        self.open_file.grid(column=4, row=1, sticky=(tk.W, tk.S, tk.E, tk.N))
        # textos
        # Discovering status text
        self.text_discover = tk.Text(left_upperframe, borderwidth=3, relief="sunken",
bg="lavender")
        self.text_discover.config (font=("consolas", 10), undo=True, wrap ='word')
        self.text_discover.config(state="normal")
        self.text_discover.insert(tk.INSERT, '\nPress "Search Nodes" to find remote XBees \n')
        self.text_discover.config(state=tk.DISABLED)
        self.text_discover.grid(row=0, column=0, sticky=(tk.W, tk.S, tk.E, tk.N))
        yscrollbar = tk.Scrollbar(left_upperframe, command=self.text_discover.yview)
        yscrollbar.grid(row=0, column=1, sticky=tk.N + tk.S)
        self.text_discover['yscrollcommand'] = yscrollbar.set
        # Texto Xbee parámetros
        self.text_params = tk.Text(right_upperframe, borderwidth=3, relief="sunken",
bg="lavender")
        self.text_params.config(font=("consolas", 10), undo=True, wrap='word')
        self.text_params.config(state="normal")
        self.text_params.insert(tk.INSERT, '\n Please press "Read local conf." to read local
node params\n'
                                           ' or put address and press "Read remote conf." to
read remote node params')
        self.text_params.config(state=tk.DISABLED)
        self.text_params.grid(row=0, column=0, sticky=(tk.W, tk.S, tk.E, tk.N))
        yscrollbar = tk.Scrollbar(right_upperframe, command=self.text_params.yview)
        yscrollbar.grid(row=0, column=1, sticky=tk.N + tk.S)
        self.text_params['yscrollcommand'] = yscrollbar.set

        # Entry address text
        self.lbladdress = tk.Label(right_bottomframe)
        self.lbladdress['text']="Remote Address: "
        #self.lbladdress.config(0,weight=1)
        self.lbladdress.grid(column=0, row=0,sticky=(tk.N, tk.W, tk.E, tk.S))
        self.remote_address = tk.Entry(right_bottomframe)
        #self.address.config(0,weight=3)
        self.remote_address.grid(column=1, row=0,sticky=(tk.N, tk.W, tk.E, tk.S))



if __name__ == "__main__":
    root = tk.Tk()
    app = Application(master=root)
    app.mainloop()
```

12.2.4 read_local_params_xbee.py

```
# Sergio Rueda Teruel. 2018
# Este software ha sido desarrollado para el trabajo fin de master de la titulación
# Máster Universitario en Ingeniería de Telecomunicación UOC-URL de la
# Universidad Oberta de Catalunya y lleva por título
# "Diseño de una WSN para la estimación del seeing de la cúpula D080,
# en el Observatorio Astrofísico de Javalambre."
# Para la realización de este código se han utilizado las librerías Python
# que la empresa Digi (Digi International Inc.) proporciona en su página web
# (https://www.digi.com/blog/xbee/introducing-the-official-digi-xbee-python-library/)
# este código está sometido a licencia de Reconocimiento-NoComercial-CompartirIgual
# 3.0 España de Creative Commons.

# Este módulo se utiliza para la lectura de los parámetros de configuración a través del
# puerto serie del nodo local. Muestra los datos por pantalla y crea un log que puede ser
# recogido por otros módulos para su visualización. (Módulo gui.py en este proyecto)


from digi.xbee.devices import XBeeDevice
from digi.xbee.util import utils
import sys
import read_sys_config
sys.tracebacklimit = 1

def main():
```

```
# Parámetros de conexión con el puerto serie al dispositivo local
port = read_sys_config.ReadLocalPortFromFile()
baud_rate = read_sys_config.ReadLocalBaudRateFromFile()
local_device = XBeeDevice(port, baud_rate)


try:

    print(" +-----------------------------------------------+")
    print(" |         Get Local XBee Parameters             |")
    print(" +-----------------------------------------------+\n")

    local_device.open()
   # Get Hardware Models with extended DIO (P5 to P9)
    Hardware_Extended = read_sys_config.ReadHardwareVersionWhithP5ToP9PinsFromFile()
   # Get parameters.
    # Diagnostic Commads
    VR = utils.hex_to_string(local_device.get_parameter("VR"))
    HV = utils.hex_to_string(local_device.get_parameter("HV"))
    AI = utils.hex_to_string(local_device.get_parameter("AI"))
    DB = utils.hex_to_string(local_device.get_parameter("DB"))
    V =  utils.hex_to_string(local_device.get_parameter("%V"))
    # Networking
    ID = utils.hex_to_string(local_device.get_parameter("ID"))
    SC = utils.hex_to_string(local_device.get_parameter("SC"))
    SD = utils.hex_to_string(local_device.get_parameter("SD"))
    ZS = utils.hex_to_string(local_device.get_parameter("ZS"))
    NJ = utils.hex_to_string(local_device.get_parameter("NJ"))
    NW = utils.hex_to_string(local_device.get_parameter("NW"))
    JV = utils.hex_to_string(local_device.get_parameter("JV"))
    JN = utils.hex_to_string(local_device.get_parameter("JN"))
    OP = utils.hex_to_string(local_device.get_parameter("OP"))
    OI = utils.hex_to_string(local_device.get_parameter("OI"))
    CH = utils.hex_to_string(local_device.get_parameter("CH"))
    NC = utils.hex_to_string(local_device.get_parameter("NC"))
    CE = utils.hex_to_string(local_device.get_parameter("CE"))
    DO = utils.hex_to_string(local_device.get_parameter("DO"))
    DC = utils.hex_to_string(local_device.get_parameter("DC"))
    # Addressing
    SH = utils.hex_to_string(local_device.get_parameter("SH"))
    SL = utils.hex_to_string(local_device.get_parameter("SL"))
    MY = utils.hex_to_string(local_device.get_parameter("MY"))
    MP = utils.hex_to_string(local_device.get_parameter("MP"))
    DH = utils.hex_to_string(local_device.get_parameter("DH"))
    DL = utils.hex_to_string(local_device.get_parameter("DL"))
    NI = local_device.get_parameter("NI").decode()
    NH = utils.hex_to_string(local_device.get_parameter("NH"))
    BH = utils.hex_to_string(local_device.get_parameter("BH"))
    AR = utils.hex_to_string(local_device.get_parameter("AR"))
    DD = utils.hex_to_string(local_device.get_parameter("DD"))
    NT = utils.hex_to_string(local_device.get_parameter("NT"))
    NO = utils.hex_to_string(local_device.get_parameter("NO"))
    NP = utils.hex_to_string(local_device.get_parameter("NP"))
    CR = utils.hex_to_string(local_device.get_parameter("CR"))
    # ZigBee Addressing
    SE = utils.hex_to_string(local_device.get_parameter("SE"))
    DE = utils.hex_to_string(local_device.get_parameter("DE"))
    CI = utils.hex_to_string(local_device.get_parameter("CI"))
    TO = utils.hex_to_string(local_device.get_parameter("TO"))
    # RF Interfacing
    PL = utils.hex_to_string(local_device.get_parameter("PL"))
    PM = utils.hex_to_string(local_device.get_parameter("PM"))
    PP = utils.hex_to_string(local_device.get_parameter("PP"))
    # Security
    EE = utils.hex_to_string(local_device.get_parameter("EE"))
    EO = utils.hex_to_string(local_device.get_parameter("EO"))
    KY = utils.hex_to_string(local_device.get_parameter("KY"))
    NK = utils.hex_to_string(local_device.get_parameter("NK"))
    # Serial Interfacing
    BD = utils.hex_to_string(local_device.get_parameter("BD"))
    NB = utils.hex_to_string(local_device.get_parameter("NB"))
    SB = utils.hex_to_string(local_device.get_parameter("SB"))
    RO = utils.hex_to_string(local_device.get_parameter("RO"))
    D6 = utils.hex_to_string(local_device.get_parameter("D6"))
    D7 = utils.hex_to_string(local_device.get_parameter("D7"))
    AP = utils.hex_to_string(local_device.get_parameter("AP"))
    AO = utils.hex_to_string(local_device.get_parameter("AO"))
```

```
# AT Command Options
CT = utils.hex_to_string(local_device.get_parameter("CT"))
GT = utils.hex_to_string(local_device.get_parameter("GT"))
CC = utils.hex_to_string(local_device.get_parameter("CC"))
# Sleep Modes
SP = utils.hex_to_string(local_device.get_parameter("SP"))
SN = utils.hex_to_string(local_device.get_parameter("SN"))
SM = utils.hex_to_string(local_device.get_parameter("SM"))
ST = utils.hex_to_string(local_device.get_parameter("ST"))
SO = utils.hex_to_string(local_device.get_parameter("SO"))
WH = utils.hex_to_string(local_device.get_parameter("WH"))
PO = utils.hex_to_string(local_device.get_parameter("PO"))
# I/O Settomgs
D0 = utils.hex_to_string(local_device.get_parameter("D0"))
D1 = utils.hex_to_string(local_device.get_parameter("D1"))
D2 = utils.hex_to_string(local_device.get_parameter("D2"))
D3 = utils.hex_to_string(local_device.get_parameter("D3"))
D4 = utils.hex_to_string(local_device.get_parameter("D4"))
D5 = utils.hex_to_string(local_device.get_parameter("D5"))
D8 = utils.hex_to_string(local_device.get_parameter("D8"))
D9 = utils.hex_to_string(local_device.get_parameter("D9"))
P0 = utils.hex_to_string(local_device.get_parameter("P0"))
P1 = utils.hex_to_string(local_device.get_parameter("P1"))
P2 = utils.hex_to_string(local_device.get_parameter("P2"))
P3 = utils.hex_to_string(local_device.get_parameter("P3"))
P4 = utils.hex_to_string(local_device.get_parameter("P4"))
if HV == Hardware_Extended:
    P5 = utils.hex_to_string(local_device.get_parameter("P5"))
    P6 = utils.hex_to_string(local_device.get_parameter("P6"))
    P7 = utils.hex_to_string(local_device.get_parameter("P7"))
    P8 = utils.hex_to_string(local_device.get_parameter("P8"))
    P9 = utils.hex_to_string(local_device.get_parameter("P9"))
PR = utils.hex_to_string(local_device.get_parameter("PR"))
PD = utils.hex_to_string(local_device.get_parameter("PD"))
LT = utils.hex_to_string(local_device.get_parameter("LT"))
RP = utils.hex_to_string(local_device.get_parameter("RP"))
# I/O Sampling
IR = utils.hex_to_string(local_device.get_parameter("IR"))
IC = utils.hex_to_string(local_device.get_parameter("IC"))
Vplus = utils.hex_to_string(local_device.get_parameter("V+"))

#print parameters
log= " +------------------------------+\n"
log= log + " | Networking                   |\n"
log = log + " +------------------------------+\n"
log = log + " PAN ID:                        %s" % ID + "\n"
log = log + " Scan Channel:                  %s" % SC + "\n"
log = log + " Scan Duration:                 %s" % SD + "\n"
log = log + " Network Watchdog Timeout:      %s" % NW + "\n"
log = log + " Channel Verification:          %s" % JV + "\n"
log = log + " Join Notification:             %s" % JN + "\n"
log = log + " Operating PAN ID:              %s" % OP + "\n"
log = log + " Operating 16-bit PAN ID:       %s" % OI + "\n"
log = log + " Operating Channel:             %s" % CH + "\n"
log = log + " Number of Remaining Children:%s" % NC + "\n"
log = log + " Coordinator Enable:            %s" % CE + "\n"
log = log + " Device Options:                %s" % DO + "\n"
log = log + " Device Controls:               %s" % DC + "\n\n"

log = log + " +------------------------------+\n"
log = log + " | Addressing                    |\n"
log = log + " +------------------------------+\n"
log = log + " Serial Number High:            %s" % SH + "\n"
log = log + " Serial Number Low:             %s" % SL + "\n"
log = log + " 16-bit Network Address:        %s" % MY + "\n"
log = log + " 16-bit Parent Address:         %s" % MP + "\n"
log = log + " Destination Address High:      %s" % DH + "\n"
log = log + " Destination Address Low:       %s" % DL + "\n"
log = log + " Node Identifier:               %s" % NI + "\n"
log = log + " Maximum Hops:                  %s" % NH + "\n"
log = log + " Broadcast Radius:              %s" % BH + "\n"
log = log + " Many-to-One Route Bro. Time: %s" % AR + "\n"
log = log + " Device Type Identifier:        %s" % DD + "\n"
log = log + " Node Discovery Backoff:        %s" % NT + "\n"
log = log + " Node Discovery Options:        %s" % NO + "\n"
log = log + " Maximum Num.Trans. Bytes:      %s" % NP + "\n"
log = log + " PAN Conflict Threshold:        %s" % CR + "\n\n"
```

```
log = log + " +------------------------------+\n"
log = log + " | ZigBee Addressing             |\n"
log = log + " +------------------------------+\n"
log = log + " Zigbee Source Endpoint:      %s" % SE + "\n"
log = log + " Zigbee Destination Endpoint: %s" % DE + "\n"
log = log + " Zigbee Cluster ID:           %s" % CI + "\n"
log = log + " Transmit Options:            %s" % TO + "\n\n"

log = log + " +------------------------------+\n"
log = log + " | RF Interfacing                |\n"
log = log + " +------------------------------+\n"
log = log + " Tx Power Level:              %s" % PL + "\n"
log = log + " Power Mode:                  %s" % PM + "\n"
log = log + " Power at PL4:                %s" % PP + "\n\n"

log = log + " +------------------------------+\n"
log = log + " | Security                      |\n"
log = log + " +------------------------------+\n"
log = log + " Encryption Enable:           %s" % EE + "\n"
log = log + " Encryption Options:          %s" % EO + "\n"
log = log + " Encryption Key:              %s" % KY + "\n"
log = log + " Network Encryption Key:      %s" % NK + "\n\n"

log = log + " +------------------------------+\n"
log = log + " | Serial Interfacing            |\n"
log = log + " +------------------------------+\n"
log = log + " Baud Rate:                   %s" % BD + "\n"
log = log + " Parity:                      %s" % NB + "\n"
log = log + " Stop Bits:                   %s" % SB + "\n"
log = log + " Packetization Timeout:       %s" % RO + "\n"
log = log + " DIO6/nRTS Configuration  :   %s" % D6 + "\n"
log = log + " DIO7/nCTS Configuration:     %s" % D7 + "\n"
log = log + " API Enable:                  %s" % AP + "\n"
log = log + " API Output Mode:             %s" % AO + "\n\n"

log = log + " +------------------------------+\n"
log = log + " | AT Command Options            |\n"
log = log + " +------------------------------+\n"
log = log + " AT Command Mode Timeout:     %s" % CT + "\n"
log = log + " Guard Times:                 %s" % GT + "\n"
log = log + " Command Sequence Character:  %s" % CC + "\n\n"

log = log + " +------------------------------+\n"
log = log + " | Sleep Modes                   |\n"
log = log + " +------------------------------+\n"
log = log + " Cyclic Sleep Period:         %s" % SP + "\n"
log = log + " Number of Cyclic Sleep Per.: %s" % SN + "\n"
log = log + " Sleep Mode:                  %s" % SM + "\n"
log = log + " Time Before Sleep:sss        %s" % ST + "\n"
log = log + " Sleep Options:               %s" % SO + "\n"
log = log + " Wake Hosts:                  %s" % WH + "\n"
log = log + " Poll Rate:                   %s" % PO + "\n\n"

log = log + " +------------------------------+\n"
log = log + " | I/O Settings                  |\n"
log = log + " +------------------------------+\n"
log = log + " DIO0/ADO/CB Configuration:   %s" % D0 + "\n"
log = log + " DIO1/AD1 Configuration:      %s" % D1 + "\n"
log = log + " DIO2/AD2 Configuration:      %s" % D2 + "\n"
log = log + " DIO3/AD3 Configuration:      %s" % D3 + "\n"
log = log + " DIO4 Configuration:          %s" % D4 + "\n"
log = log + " DIO5 Configuration:          %s" % D5 + "\n"
log = log + " DIO8 Configuration:          %s" % D8 + "\n"
log = log + " DIO9 Configuration:          %s" % D9 + "\n"
log = log + " DIO10 Configuration:         %s" % P0 + "\n"
log = log + " DIO11 Configuration:         %s" % P1 + "\n"
log = log + " DIO12 Configuration:         %s" % P2 + "\n"
log = log + " DIO13 Configuration:         %s" % P3 + "\n"
log = log + " DIO14 Configuration:         %s" % P4 + "\n"
if HV == Hardware_Extended:
    log = log + " DIO15 Configuration:         %s" % P5 + "\n"
    log = log + " DIO16 Configuration:         %s" % P6 + "\n"
    log = log + " DIO17 Configuration:         %s" % P7 + "\n"
    log = log + " DIO18 Configuration:         %s" % P8 + "\n"
    log = log + " DIO19 Configuration:         %s" % P9 + "\n"
log = log + " Pull-UP Resistor Enable:     %s" % PR + "\n"
```

```
            log = log + " Pull-Up/down Direction:       %s" % PD + "\n"
            log = log + " Associate LED Blink Time:     %s" % LT + "\n"
            log = log + " RSSI PWM Timer:               %s" % RP + "\n\n"

            log = log + " +-----------------------------+\n"
            log = log + " | I/O Sampling                |\n"
            log = log + " +-----------------------------+\n"
            log = log + " IO Sampling Rate:             %s" % IR + "\n"
            log = log + " Digital IO Change Detection: %s" % IC + "\n"
            log = log + " Supply Votage High Thres.:    %s" % Vplus + "\n\n"

            log = log + " +-----------------------------+\n"
            log = log + " | Diagnostic Commands         |\n"
            log = log + " +-----------------------------+\n"
            log = log + " Firmaware Version:            %s" % VR + "\n"
            log = log + " Hardware Version:             %s" % HV + "\n"
            log = log + " Association Indication:       %s" % AI + "\n"
            log = log + " RSSI of Last Packet:          %s" % DB + "\n"
            log = log + " Supply Votage:                %s" % V + "\n\n"
            print (log)

        except:
            if local_device.is_open():
                local_device.close()
            log ="Sorry an error has happened"
            pass


        finally:
            if local_device is not None and local_device.is_open():
                local_device.close()
            pass


        return log

if __name__ == '__main__':
    main()
```

12.2.5 read_node_config_file.py

```
# Sergio Rueda Teruel. 2018
# Este software ha sido desarrollado para el trabajo fin de máster de la titulación
# Máster Universitario en Ingeniería de Telecomunicación UOC-URL de la
# Universidad Oberta de Catalunya y lleva por título
# "Diseño de una WSN para la estimación del seeing de la cúpula D080,
# en el Observatorio Astrofísico de Javalambre."
# Este código está sometido a licencia de Reconocimiento-NoComercial-CompartirIgual
# 3.0 España de Creative Commons.

# Este módulo incorpora una serie de funciones que son llamadas por otros módulos para
# la lectura de los parámetros de configuración desde unos archivos tipo ini.
# Existirá uno de estos archivos ini por nodo que queramos configurar, y el módulo de
# configuración utilizará las funciones aquí desarrolladas para parsear los datos del
# fichero a valores que pueda utilizar en la configuración



import configparser

config_file = "config/"
settings = configparser.ConfigParser()



def set_path_to_file(path):
    global config_file
    config_file = "config/" + path +".ini"
    settings.read(config_file)



# Obtenemos del fichero de configuración los valores para cada parámetro
```

```
# Networking
def ReadPanIDFromFile(Address):
    return settings.get(Address, "ID")


def ReadScanChannelsFromFile(Address):
    print ("aqui")
    return settings.get(Address, "SC")


def ReadScanDurationFromFile(Address):
    return settings.get(Address, "SD")


def ReadZigBeeStackProfileFromFile(Address):
    return settings.get(Address, "ZS")


def ReadNodeJoinTimeFromFile(Address):
    return settings.get(Address, "NJ")


def ReadNetworkWatchdogTimeoutFromFile(Address):
    return settings.get(Address, "NW")


def ReadChannelVerificationFromFile(Address):
    return settings.get(Address, "JV")


def ReadJoinNotificationFromFile(Address):
    return settings.get(Address, "JN")


def ReadCoordinatorEnableFromFile(Address):
    return settings.get(Address, "CE")


def ReadDeviceOptionsFromFile(Address):
    return settings.get(Address, "DO")


def ReadDeviceControlsFromFile(Address):
    return settings.get(Address, "DC")


# Addressing
def ReadDestinationAddressHighFromFile(Address):
    return settings.get(Address, "DH")


def ReadDestinationAddressLowFromFile(Address):
    return settings.get(Address, "DL")


def ReadNodeIdentifierFromFile(Address):
    return settings.get(Address, "NI")


def ReadMaximumHopsFromFile(Address):
    return settings.get(Address, "NH")


def ReadBroadcastRadiusFromFile(Address):
    return settings.get(Address, "BH")


def ReadManyToOneRouteBroadcastTimeFromFile(Address):
    return settings.get(Address, "AR")


def ReadDeviceTypeIdentifierFromFile(Address):
    return settings.get(Address, "DD")


def ReadNodeDiscoveryBackoffFromFile(Address):
    return settings.get(Address, "NT")
```

```
def ReadNodeDiscoveryOptionsFromFile(Address):
    return settings.get(Address, "NO")


def ReadPanConflictThresholdFromFile(Address):
    return settings.get(Address, "CR")


# ZigBee Addressing
def ReadZigBeeSourceEndPointFromFile(Address):
    return settings.get(Address, "SE")


def ReadZigBeeDestinationEndpointFromFile(Address):
    return settings.get(Address, "DE")


def ReadZigBeeClusterIDFromFile(Address):
    return settings.get(Address, "CI")


def ReadTransmitOptionsFromFile(Address):
    return settings.get(Address, "TO")


# RF Interfacing
def ReadTxPowerLevelFromFile(Address):
    return settings.get(Address, "PL")


def ReadPowerModeFromFile(Address):
    return settings.get(Address, "PM")


# Security
def ReadEncryptionEnableFromFile(Address):
    return settings.get(Address, "EE")


def ReadEncryptionOptionsFromFile(Address):
    return settings.get(Address, "EO")


def ReadEncryptionKeyFromFile(Address):
    return settings.get(Address, "KY")


def ReadNetworkEncryptionKeyFromFile(Address):
    return settings.get(Address, "NK")


# Serial Interfacing
def ReadBaudRateFromFile(Address):
    return settings.get(Address, "BD")


def ReadParityFromFile(Address):
    return settings.get(Address, "NB")


def ReadStopBitsFromFile(Address):
    return settings.get(Address, "SB")


def ReadPacketizationTimeoutFromFile(Address):
    return settings.get(Address, "RO")


def ReadDIO6ConfigurationFromFile(Address):
    return settings.get(Address, "D6")


def ReadDIO7ConfigurationFromFile(Address):
    return settings.get(Address, "D7")
```

```
def ReadApiEnableFromFile(Address):
    return settings.get(Address, "AP")


def ReadApiOutputModeFromFile(Address):
    return settings.get(Address, "AO")


# AT Command
def ReadATCommandModeTimeoutFromFile(Address):
    return settings.get(Address, "CT")


def ReadGuardTimesFromFile(Address):
    return settings.get(Address, "GT")


def ReadCommandSequenceCharacterFromFile(Address):
    return settings.get(Address, "CC")


# Sleep Modes
def ReadCyclicSleepPeriodFromFile(Address):
    return settings.get(Address, "SP")


def ReadNumberOfCyclicSleepPeriodsFromFile(Address):
    return settings.get(Address, "SN")


def ReadSleepModeFromFile(Address):
    return settings.get(Address, "SM")


def ReadTimeBeforeSleepFromFile(Address):
    return settings.get(Address, "ST")


def ReadSleepOptionsFromFile(Address):
    return settings.get(Address, "SO")


def ReadWakeHostFromFile(Address):
    return settings.get(Address, "WH")


def ReadPollRateFromFile(Address):
    return settings.get(Address, "PO")


# I/O Settings
def ReadDIO0AD0ConfigurationFromFile(Address):
    return settings.get(Address, "D0")


def ReadDIO1AD1ConfigurationFromFile(Address):
    return settings.get(Address, "D1")


def ReadDIO2AD2ConfigurationFromFile(Address):
    return settings.get(Address, "D2")


def ReadDIO3AD3ConfigurationFromFile(Address):
    return settings.get(Address, "D3")


def ReadDIO4ConfigurationFromFile(Address):
    return settings.get(Address, "D4")


def ReadDIO5ConfigurationFromFile(Address):
    return settings.get(Address, "D5")


def ReadDIO8ConfigurationFromFile(Address):
```

```
    return settings.get(Address, "D8")


def ReadDIO9ConfigurationFromFile(Address):
    return settings.get(Address, "D9")


def ReadDIO10ConfigurationFromFile(Address):
    return settings.get(Address, "P0")


def ReadDIO11ConfigurationFromFile(Address):
    return settings.get(Address, "P1")


def ReadDIO12ConfigurationFromFile(Address):
    return settings.get(Address, "P2")


def ReadDIO13ConfigurationFromFile(Address):
    return settings.get(Address, "P3")


def ReadDIO14ConfigurationFromFile(Address):
    return settings.get(Address, "P4")

def ReadDIO15ConfigurationFromFile(Address):
    return settings.get(Address, "P5")

def ReadDIO16ConfigurationFromFile(Address):
    return settings.get(Address, "P6")

def ReadDIO17ConfigurationFromFile(Address):
    return settings.get(Address, "P7")

def ReadDIO18ConfigurationFromFile(Address):
    return settings.get(Address, "P8")

def ReadDIO19ConfigurationFromFile(Address):
    return settings.get(Address, "P9")


def ReadPullUpResistorEnableFromFile(Address):
    return settings.get(Address, "PR")


def ReadPullUpDownDirectionFromFile(Address):
    return settings.get(Address, "PD")


def ReadAssociatedLedBlinkTimeFromFile(Address):
    return settings.get(Address, "LT")


def ReadRssiPwmTimerFromFile(Address):
    return settings.get(Address, "RP")

# I/O Sampling
def ReadIOSamplingRateFromFile(Address):
    return settings.get(Address, "IR")


def ReadDigitalIOChangeDetectionFromFile(Address):
    return settings.get(Address, "IC")


def ReadSupplyVoltageHihgThresholdFromFile(Address):
    return settings.get(Address, "V+")



def main():
    try:
        settings.read(config_file)

    except:
        print("error opening config.ini")
```

```
if __name__ == '__main__':
    main()
```

12.2.6 read_node_list.py

```
# Sergio Rueda Teruel. 2018
# Este software ha sido desarrollado para el trabajo fin de máster de la titulación
# Máster Universitario en Ingeniería de Telecomunicación UOC-URL de la
# Universidad Oberta de Catalunya y lleva por título
# "Diseño de una WSN para la estimación del seeing de la cúpula D080,
# en el Observatorio Astrofísico de Javalambre."
# Este código está sometido a licencia de Reconocimiento-NoComercial-CompartirIgual
# 3.0 España de Creative Commons.

# Este módulo incorpora una función que es llamada por otros módulos para
# determinar el número de nodos que queramos encuestar para la aquisición
# de datos y el NI (Node identificator) de cada uno de ellos.

CONFIG_FILE = "config/list_of_nodes.ini"

def ReadListOfNodesFromFile():
    with open(CONFIG_FILE, "r") as ins:
        array = []
        for line in ins:
            array.append(line.strip('\n'))
    return array


if __name__ == '__main__':
    ReadListOfNodesFromFile()
```

12.2.7 read_remote_ADC_xbee_multihilo.py

```
# Sergio Rueda Teruel. 2018
# Este software ha sido desarrollado para el trabajo fin de master de la titulación
# Máster Universitario en Ingeniería de Telecomunicación UOC-URL de la
# Universidad Oberta de Catalunya y lleva por título
# "Diseño de una WSN para la estimación del seeing de la cúpula D080,
# en el Observatorio Astrofísico de Javalambre."
# Para la realización de este código se han utilizado las librerías Python
# que la empresa Digi (Digi International Inc.) proporciona en su página web
# (https://www.digi.com/blog/xbee/introducing-the-official-digi-xbee-python-library/)
# este código está sometido a licencia de Reconocimiento-NoComercial-CompartirIgual
# 3.0 España de Creative Commons.

# Este módulo se utiliza para la lectura de los valores de las entradas de los nodos
# xbee remotos periódicamente, para ello se crea un hilo por cada nodo y se les va
# encuestando, para mayor estabilidad cada nodo hilo bloquea a los demas y así no se
# desborda el buffer del nodo colector.
# Antes de empezar a encuestar a los nodos se realiza un procedimiento de descubrimiento
# de cada uno de los nodos listados en el archivo de configuración list_of_nodes.ini
# Si un nodo no responde a una petición se le empieza a encuestar a mayor frecuencia
# para determinar si ha sido un fallo temporal o un fallo permanente, en caso de que el
# número de fallos seguidos supere un cierto valor hace que se considere al nodo en fallo
# y se deje de encuestarlo.


from digi.xbee.devices import XBeeDevice
from digi.xbee.io import IOLine, IOMode
from digi.xbee.util import utils
from digi.xbee.exception import TimeoutException

import logging
import sys
import threading
import time

from datetime import datetime

import read_sys_config
import read_node_list
```

```
import send_data_to_telegraf


REMOTE_NODES_ID = read_node_list.ReadListOfNodesFromFile()
MAX_INTENTOS_DESCUBRIMIENTO = 3 # Número de intentos de descubrimiento de cada nodo
MAX_INTENTOS_LEER_DATOS = 20 # Número máximo de fallos seguidos permitidos al leer
                             # datos de un nodo.
# Definición lineas de entrada
IOLINE_IN_3 = IOLine.DIO3_AD3
IOLINE_IN_2 = IOLine.DIO2_AD2
IOLINE_IN_1 = IOLine.DIO1_AD1
IOLINE_IN_0 = IOLine.DIO0_AD0
# Tensión máxima en las entradas (mv)
MAX_VOLTAGE_INPUT = 1200
# Frecuencia de encuesta a los nodos en caso de que NO existan problemas (sg)
LONG_WAIT=30
# Frecuencia de encuesta a los nodos en caso de que SI existan problemas (sg)
SHORT_WAIT=1

# Parámetros de conexión con el puerto serie al dispositivo local
port = read_sys_config.ReadLocalPortFromFile()
baud_rate = read_sys_config.ReadLocalBaudRateFromFile()

# Función para calcular el valor de temperatura medainte el valor crudo de la
# tensión de entrada y voltaje de alimentación (divisor de tensión)
def ntc10k_calculate_temp(raw_value, volts):
    # mV
    voltage = (MAX_VOLTAGE_INPUT * raw_value / 1024)
    Rntc = 210000 * (voltage / (volts - voltage))
    # AJUSTE_CUADRÁTICO (según curva calculada en Matlab)
    a=574.1
    b=-0.1242
    c= -158
    temperatureC =a*(pow(Rntc,b))+c
    logging.debug('ntc 10K temper: %.2f',temperatureC)
    return temperatureC



def main():
    class ReadAD:
        def __init__(self, start=0):
            self.lock = threading.Lock()
            self.value = start


        def read_AD(self, indice):
            logging.debug('Waiting for lock')
            self.lock.acquire()
            try:
                self.timeout = False
                logging.debug('Acquired lock')
                vcc = nodos_activos[indice].get_parameter("%V")
                vcc = int(utils.hex_to_string(vcc).replace(' ', ''), 16)
                # Leemos el valor crudo de las entradas analógicas
                raw_value_1 = nodos_activos[indice].get_adc_value(IOLINE_IN_0)
                raw_value_2 = nodos_activos[indice].get_adc_value(IOLINE_IN_1)
                raw_value_3 = nodos_activos[indice].get_adc_value(IOLINE_IN_2)
                raw_value_4 = nodos_activos[indice].get_adc_value(IOLINE_IN_3)

                # Calculamos el valor de temperatura en cada entrada en función de la tensión
de alimentación y del
                tntc_1 = ntc10k_calculate_temp(raw_value_1, vcc)
                tntc_2 = ntc10k_calculate_temp(raw_value_2, vcc)
                tntc_3 = ntc10k_calculate_temp(raw_value_3, vcc)
                tntc_4 = ntc10k_calculate_temp(raw_value_4, vcc)

                # ************************************************************************
                # ESTA ES LA PARTE DE TELEGRAF
                send_data_to_telegraf.main(REMOTE_NODES_ID[indice], tntc_1, tntc_2, tntc_3,
tntc_4, float(vcc))


            except TimeoutException:
                self.timeout = True
                logging.debug('ADC error')
                local_device.reset()
```

```
            finally:
                self.lock.release()

            return self.timeout

# función que realiza las tareas de lectura de las entradas en los nodos
    timeouts=[]
    def worker(c,i):
        #La petición se hace con tiempo variable, si no responde, esto es si da timeout se
hacen más rápidas para ver
        #si el nodo estaba en una secuencia de sueño, si sobre pasa el límite de timeouts
cortos ya no se le pregunta más
        timeouts.append(0)
        try:
            while True and (timeouts[i]< MAX_INTENTOS_LEER_DATOS):
                logging.debug('Stamp: %s',  str(datetime.now()))
                if c.read_AD(i):
                    timeouts[i]+=1
                    logging.debug("Timeouts %s", timeouts)
                    pause=SHORT_WAIT
                else:
                    timeouts[i] = 0 #reseteamos la cuenta de timeouts
                    pause = LONG_WAIT
                logging.debug('Sleeping %0.02f', pause)
                time.sleep(pause)
        except ValueError:
             logging.debug('Worker error')

# función que realiza el procedimiento de descubrimiento de los nodos listado en el archivo
list_of_nodes.ini
    nodos_activos=[]
    def descubre_nodos():
        index_devices=0
        try:
            for index in range (0, len(REMOTE_NODES_ID)):
                nodo_descubierto=False
                intentos_descubir=0
                while (nodo_descubierto != True)and intentos_descubir<
MAX_INTENTOS_DESCUBRIMIENTO:
                    remote_device=(xbee_network.discover_device(REMOTE_NODES_ID[index]))
                    if remote_device is None:
                        logging.debug('Could not find the remote device: %s',
REMOTE_NODES_ID[index])
                        intentos_descubir+=1
                        logging.debug("Nodo: %s" , (REMOTE_NODES_ID[index]))
                        logging.debug('Intentos descubrimiento restantes: %s',
(MAX_INTENTOS_DESCUBRIMIENTO-intentos_descubir))
                        time.sleep(1)
                    else:
                        nodos_activos.append(remote_device)
                        index_devices+=1
                        logging.debug('Descubierto: %s',remote_device)
                        nodo_descubierto = True
        except:
            logging.debug('Error proceso descubrimiento')


# Configuracíon de la salida del log
    logging.basicConfig(
        level=logging.DEBUG,
        format='(%(threadName)-10s) %(message)s',
    )

# Conexión al nodo local
    local_device = XBeeDevice(port, baud_rate)
    xbee_network = local_device.get_network()
    local_device.open()
    descubre_nodos()

    try:
        lectura = ReadAD()
        # Creación de un hilo por cada nodo activo
        for i in range(len(nodos_activos)):
            logging.debug('creando hilo')
            t = threading.Thread(name=nodos_activos[i], target=worker, args=(lectura, i,))
            t.start()
```

```
        if len(nodos_activos)==0:
            logging.debug('No nodes found')
            sys.exti(-1)
        else:
            logging.debug('Waiting for worker threads')
            main_thread = threading.main_thread()
            for t in threading.enumerate():
                if t is not main_thread:
                    t.join()
            logging.debug('Counter: %d', lectura.value)

    except:
        logging.debug('exept')
        sys.exit(1)


if __name__ == '__main__':
    main()
```

12.2.8 read_remote_params_xbee.py

```
# Sergio Rueda Teruel. 2018
# Este software ha sido desarrollado para el trabajo fin de máster de la titulación
# Máster Universitario en Ingeniería de Telecomunicación UOC-URL de la
# Universidad Oberta de Catalunya y lleva por título
# "Diseño de una WSN para la estimación del seeing de la cúpula D080,
# en el Observatorio Astrofísico de Javalambre."
# Para la realización de este código se han utilizado las librerías Python
# que la empresa Digi (Digi International Inc.) proporciona en su página web
# (https://www.digi.com/blog/xbee/introducing-the-official-digi-xbee-python-library/)
# este código está sometido a licencia de Reconocimiento-NoComercial-CompartirIgual
# 3.0 España de Creative Commons.

# Este módulo se utiliza para la lectura de los parámetros de configuración a través de
# RF de un nodo remoto, dicha lectura se hace a través del nodo local.
# Los parámetros se muestran por pantalla y se crea un log que puede ser
# recogido por otros módulos para su visualización. (Módulo gui.py en este proyecto)

from digi.xbee.devices import XBeeDevice
from digi.xbee.devices import RemoteZigBeeDevice
from digi.xbee.util import utils
from digi.xbee.models.address import XBee64BitAddress
import sys
import read_sys_config
sys.tracebacklimit = 0

# Valor del nodo remoto por defecto si no le pasamos otro valor
PARAM_VALUE_REMOTE_NODE_ADDR="0013A200416299FE"


# Parámetros de conexión con el puerto serie al dispositivo local
PORT=read_sys_config.ReadLocalPortFromFile()
BAUD_RATE = read_sys_config.ReadLocalBaudRateFromFile()



def main(NodeAddress):
    if NodeAddress =="":
        NodeAddress=PARAM_VALUE_REMOTE_NODE_ADDR
    Address=XBee64BitAddress.from_hex_string(NodeAddress)

    print(" +-----------------------------------------------------+")
    print(" |  Get Remote XBee (" + NodeAddress + ") parameters       |")
    print(" +-----------------------------------------------------+\n")

    local_device = XBeeDevice(PORT, BAUD_RATE)


    try:
        local_device.open()
        remote_device = RemoteZigBeeDevice(local_device, Address)

        # Determinar que versión de hardware es el nodo para saber que parámetros
        # se pueden configurar
        Hardware_Extended = read_sys_config.ReadHardwareVersionWhithP5ToP9PinsFromFile()
```

```
# Get parameters.
 # Diagnostic Commads
 VR = utils.hex_to_string(remote_device.get_parameter("VR"))
 HV = utils.hex_to_string(remote_device.get_parameter("HV"))
 AI = utils.hex_to_string(remote_device.get_parameter("AI"))
 DB = utils.hex_to_string(remote_device.get_parameter("DB"))
 V =  utils.hex_to_string(remote_device.get_parameter("%V"))
 # Networking
 ID = utils.hex_to_string(remote_device.get_parameter("ID"))
 SC = utils.hex_to_string(remote_device.get_parameter("SC"))
 SD = utils.hex_to_string(remote_device.get_parameter("SD"))
 ZS = utils.hex_to_string(remote_device.get_parameter("ZS"))
 NJ = utils.hex_to_string(remote_device.get_parameter("NJ"))
 NW = utils.hex_to_string(remote_device.get_parameter("NW"))
 JV = utils.hex_to_string(remote_device.get_parameter("JV"))
 JN = utils.hex_to_string(remote_device.get_parameter("JN"))
 OP = utils.hex_to_string(remote_device.get_parameter("OP"))
 OI = utils.hex_to_string(remote_device.get_parameter("OI"))
 CH = utils.hex_to_string(remote_device.get_parameter("CH"))
 NC = utils.hex_to_string(remote_device.get_parameter("NC"))
 CE = utils.hex_to_string(remote_device.get_parameter("CE"))
 DO = utils.hex_to_string(remote_device.get_parameter("DO"))
 DC = utils.hex_to_string(remote_device.get_parameter("DC"))
 # Addressing
 SH = utils.hex_to_string(remote_device.get_parameter("SH"))
 SL = utils.hex_to_string(remote_device.get_parameter("SL"))
 MY = utils.hex_to_string(remote_device.get_parameter("MY"))
 MP = utils.hex_to_string(remote_device.get_parameter("MP"))
 DH = utils.hex_to_string(remote_device.get_parameter("DH"))
 DL = utils.hex_to_string(remote_device.get_parameter("DL"))
 NI = remote_device.get_parameter("NI").decode()
 NH = utils.hex_to_string(remote_device.get_parameter("NH"))
 BH = utils.hex_to_string(remote_device.get_parameter("BH"))
 AR = utils.hex_to_string(remote_device.get_parameter("AR"))
 DD = utils.hex_to_string(remote_device.get_parameter("DD"))
 NT = utils.hex_to_string(remote_device.get_parameter("NT"))
 NO = utils.hex_to_string(remote_device.get_parameter("NO"))
 NP = utils.hex_to_string(remote_device.get_parameter("NP"))
 CR = utils.hex_to_string(remote_device.get_parameter("CR"))
 # ZigBee Addressing
 SE = utils.hex_to_string(remote_device.get_parameter("SE"))
 DE = utils.hex_to_string(remote_device.get_parameter("DE"))
 CI = utils.hex_to_string(remote_device.get_parameter("CI"))
 TO = utils.hex_to_string(remote_device.get_parameter("TO"))
 # RF Interfacing
 PL = utils.hex_to_string(remote_device.get_parameter("PL"))
 PM = utils.hex_to_string(remote_device.get_parameter("PM"))
 PP = utils.hex_to_string(remote_device.get_parameter("PP"))
 # Security
 EE = utils.hex_to_string(remote_device.get_parameter("EE"))
 EO = utils.hex_to_string(remote_device.get_parameter("EO"))
 KY = utils.hex_to_string(remote_device.get_parameter("KY"))
 NK = utils.hex_to_string(remote_device.get_parameter("NK"))
 # Serial Interfacing
 BD = utils.hex_to_string(remote_device.get_parameter("BD"))
 NB = utils.hex_to_string(remote_device.get_parameter("NB"))
 SB = utils.hex_to_string(remote_device.get_parameter("SB"))
 RO = utils.hex_to_string(remote_device.get_parameter("RO"))
 D6 = utils.hex_to_string(remote_device.get_parameter("D6"))
 D7 = utils.hex_to_string(remote_device.get_parameter("D7"))
 AP = utils.hex_to_string(remote_device.get_parameter("AP"))
 AO = utils.hex_to_string(remote_device.get_parameter("AO"))
 # AT Command Options
 CT = utils.hex_to_string(remote_device.get_parameter("CT"))
 GT = utils.hex_to_string(remote_device.get_parameter("GT"))
 CC = utils.hex_to_string(remote_device.get_parameter("CC"))
 # Sleep Modes
 SP = utils.hex_to_string(remote_device.get_parameter("SP"))
 SN = utils.hex_to_string(remote_device.get_parameter("SN"))
 SM = utils.hex_to_string(remote_device.get_parameter("SM"))
 ST = utils.hex_to_string(remote_device.get_parameter("ST"))
 SO = utils.hex_to_string(remote_device.get_parameter("SO"))
 WH = utils.hex_to_string(remote_device.get_parameter("WH"))
 PO = utils.hex_to_string(remote_device.get_parameter("PO"))
 # I/O Settomgs
 D0 = utils.hex_to_string(remote_device.get_parameter("D0"))
 D1 = utils.hex_to_string(remote_device.get_parameter("D1"))
```

```
D2 = utils.hex_to_string(remote_device.get_parameter("D2"))
D3 = utils.hex_to_string(remote_device.get_parameter("D3"))
D4 = utils.hex_to_string(remote_device.get_parameter("D4"))
D5 = utils.hex_to_string(remote_device.get_parameter("D5"))
D8 = utils.hex_to_string(remote_device.get_parameter("D8"))
D9 = utils.hex_to_string(remote_device.get_parameter("D9"))
P0 = utils.hex_to_string(remote_device.get_parameter("P0"))
P1 = utils.hex_to_string(remote_device.get_parameter("P1"))
P2 = utils.hex_to_string(remote_device.get_parameter("P2"))
P3 = utils.hex_to_string(remote_device.get_parameter("P3"))
P4 = utils.hex_to_string(remote_device.get_parameter("P4"))
if HV == Hardware_Extended:
    P5 = utils.hex_to_string(remote_device.get_parameter("P5"))
    P6 = utils.hex_to_string(remote_device.get_parameter("P6"))
    P7 = utils.hex_to_string(remote_device.get_parameter("P7"))
    P8 = utils.hex_to_string(remote_device.get_parameter("P8"))
    P9 = utils.hex_to_string(remote_device.get_parameter("P9"))
PR = utils.hex_to_string(remote_device.get_parameter("PR"))
PD = utils.hex_to_string(remote_device.get_parameter("PD"))
LT = utils.hex_to_string(remote_device.get_parameter("LT"))
RP = utils.hex_to_string(remote_device.get_parameter("RP"))
# I/O Sampling
IR = utils.hex_to_string(remote_device.get_parameter("IR"))
IC = utils.hex_to_string(remote_device.get_parameter("IC"))
Vplus = utils.hex_to_string(remote_device.get_parameter("V+"))

# log parameters
log = " +------------------------------+\n"
log = log + " | Networking                   |\n"
log = log + " +------------------------------+\n"
log = log + " PAN ID:                      %s" % ID + "\n"
log = log + " Scan Channel:                %s" % SC + "\n"
log = log + " Scan Duration:               %s" % SD + "\n"
log = log + " Network Watchdog Timeout:    %s" % NW + "\n"
log = log + " Channel Verification:        %s" % JV + "\n"
log = log + " Join Notification:           %s" % JN + "\n"
log = log + " Operating PAN ID:            %s" % OP + "\n"
log = log + " Operating 16-bit PAN ID:     %s" % OI + "\n"
log = log + " Operating Channel:           %s" % CH + "\n"
log = log + " Number of Remaining Children:%s" % NC + "\n"
log = log + " Coordinator Enable:          %s" % CE + "\n"
log = log + " Device Options:              %s" % DO + "\n"
log = log + " Device Controls:             %s" % DC + "\n\n"

log = log + " +------------------------------+\n"
log = log + " | Addressing                   |\n"
log = log + " +------------------------------+\n"
log = log + " Serial Number High:          %s" % SH + "\n"
log = log + " Serial Number Low:           %s" % SL + "\n"
log = log + " 16-bit Network Address:      %s" % MY + "\n"
log = log + " 16-bit Parent Address:       %s" % MP + "\n"
log = log + " Destination Address High:    %s" % DH + "\n"
log = log + " Destination Address Low:     %s" % DL + "\n"
log = log + " Node Identifier:             %s" % NI + "\n"
log = log + " Maximum Hops:                %s" % NH + "\n"
log = log + " Broadcast Radius:            %s" % BH + "\n"
log = log + " Many-to-One Route Bro. Time: %s" % AR + "\n"
log = log + " Device Type Identifier:      %s" % DD + "\n"
log = log + " Node Discovery Backoff:      %s" % NT + "\n"
log = log + " Node Discovery Options:      %s" % NO + "\n"
log = log + " Maximum Num.Trans. Bytes:    %s" % NP + "\n"
log = log + " PAN Conflict Threshold:      %s" % CR + "\n\n"

log = log + " +------------------------------+\n"
log = log + " | ZigBee Addressing            |\n"
log = log + " +------------------------------+\n"
log = log + " Zigbee Source Endpoint:      %s" % SE + "\n"
log = log + " Zigbee Destination Endpoint: %s" % DE + "\n"
log = log + " Zigbee Cluster ID:           %s" % CI + "\n"
log = log + " Transmit Options:            %s" % TO + "\n\n"

log = log + " +------------------------------+\n"
log = log + " | RF Interfacing               |\n"
log = log + " +------------------------------+\n"
log = log + " Tx Power Level:              %s" % PL + "\n"
log = log + " Power Mode:                  %s" % PM + "\n"
log = log + " Power at PL4:                %s" % PP + "\n\n"
```

```
log = log + " +------------------------------+\n"
log = log + " | Security                      |\n"
log = log + " +------------------------------+\n"
log = log + " Encryption Enable:            %s" % EE + "\n"
log = log + " Encryption Options:           %s" % EO + "\n"
log = log + " Encryption Key:               %s" % KY + "\n"
log = log + " Network Encryption Key:       %s" % NK + "\n\n"

log = log + " +------------------------------+\n"
log = log + " | Serial Interfacing            |\n"
log = log + " +------------------------------+\n"
log = log + " Baud Rate:                    %s" % BD + "\n"
log = log + " Parity:                       %s" % NB + "\n"
log = log + " Stop Bits:                    %s" % SB + "\n"
log = log + " Packetization Timeout:        %s" % RO + "\n"
log = log + " DIO6/nRTS Configuration  :    %s" % D6 + "\n"
log = log + " DIO7/nCTS Configuration:      %s" % D7 + "\n"
log = log + " API Enable:                   %s" % AP + "\n"
log = log + " API Output Mode:              %s" % AO + "\n\n"

log = log + " +------------------------------+\n"
log = log + " | AT Command Options            |\n"
log = log + " +------------------------------+\n"
log = log + " AT Command Mode Timeout:      %s" % CT + "\n"
log = log + " Guard Times:                  %s" % GT + "\n"
log = log + " Command Sequence Character:   %s" % CC + "\n\n"

log = log + " +------------------------------+\n"
log = log + " | Sleep Modes                   |\n"
log = log + " +------------------------------+\n"
log = log + " Cyclic Sleep Period:          %s" % SP + "\n"
log = log + " Number of Cyclic Sleep Per.: %s" % SN + "\n"
log = log + " Sleep Mode:                   %s" % SM + "\n"
log = log + " Time Before Sleep:sss         %s" % ST + "\n"
log = log + " Sleep Options:                %s" % SO + "\n"
log = log + " Wake Hosts:                   %s" % WH + "\n"
log = log + " Poll Rate:                    %s" % PO + "\n\n"

log = log + " +------------------------------+\n"
log = log + " | I/O Settings                  |\n"
log = log + " +------------------------------+\n"
log = log + " DIO0/ADO/CB Configuration:    %s" % D0 + "\n"
log = log + " DIO1/AD1 Configuration:       %s" % D1 + "\n"
log = log + " DIO2/AD2 Configuration:       %s" % D2 + "\n"
log = log + " DIO3/AD3 Configuration:       %s" % D3 + "\n"
log = log + " DIO4 Configuration:           %s" % D4 + "\n"
log = log + " DIO5 Configuration:           %s" % D5 + "\n"
log = log + " DIO8 Configuration:           %s" % D8 + "\n"
log = log + " DIO9 Configuration:           %s" % D9 + "\n"
log = log + " DIO10 Configuration:          %s" % P0 + "\n"
log = log + " DIO11 Configuration:          %s" % P1 + "\n"
log = log + " DIO12 Configuration:          %s" % P2 + "\n"
log = log + " DIO13 Configuration:          %s" % P3 + "\n"
log = log + " DIO14 Configuration:          %s" % P4 + "\n"
if HV == Hardware_Extended:
    log = log + " DIO15 Configuration:          %s" % P5 + "\n"
    log = log + " DIO16 Configuration:          %s" % P6 + "\n"
    log = log + " DIO17 Configuration:          %s" % P7 + "\n"
    log = log + " DIO18 Configuration:          %s" % P8 + "\n"
    log = log + " DIO19 Configuration:          %s" % P9 + "\n"
log = log + " Pull-UP Resistor Enable:      %s" % PR + "\n"
log = log + " Pull-Up/down Direction:       %s" % PD + "\n"
log = log + " Associate LED Blink Time:     %s" % LT + "\n"
log = log + " RSSI PWM Timer:               %s" % RP + "\n\n"

log = log + " +------------------------------+\n"
log = log + " | I/O Sampling                  |\n"
log = log + " +------------------------------+\n"
log = log + " IO Sampling Rate:             %s" % IR + "\n"
log = log + " Digital IO Change Detection: %s" % IC + "\n"
log = log + " Supply Votage High Thres.:    %s" % Vplus + "\n\n"

log = log + " +------------------------------+\n"
log = log + " | Diagnostic Commands           |\n"
log = log + " +------------------------------+\n"
log = log + " Firmaware Version:            %s" % VR + "\n"
```

```
        log = log + " Hardware Version:             %s" % HV + "\n"
        log = log + " Association Indication:       %s" % AI + "\n"
        log = log + " RSSI of Last Packet:          %s" % DB + "\n"
        log = log + " Supply Votage:                %s" % V + "\n\n"


    except:
        log="No device found\n"
        if local_device.is_open():
            local_device.close()
        pass


    finally:
        if local_device.is_open():
            local_device.close()

    return log

if __name__ == '__main__':
    main(PARAM_VALUE_REMOTE_NODE_ADDR)
```

12.2.9 read_sys_config.py

```
# Sergio Rueda Teruel. 2018
# Este software ha sido desarrollado para el trabajo fin de máster de la titulación
# Máster Universitario en Ingeniería de Telecomunicación UOC-URL de la
# Universidad Oberta de Catalunya y lleva por título
# "Diseño de una WSN para la estimación del seeing de la cúpula D080,
# en el Observatorio Astrofísico de Javalambre."
# Este código está sometido a licencia de Reconocimiento-NoComercial-CompartirIgual
# 3.0 España de Creative Commons.

# Este módulo incorpora una serie de funciones que son llamadas por otros módulos para
# la lectura de los parámetros de configuración del puerto serie.
# También se indica que versión de hardware presenta entradas P5 a P9, ya que no todas
# las versiones de hardware lo hacen


import configparser

CONFIG_FILE = "config/sys_config.ini"

settings = configparser.ConfigParser()
settings.read(CONFIG_FILE)



def ReadHardwareVersionWhithP5ToP9PinsFromFile():
    return settings.get("COMMON", "HARDWARE_VERSION_WITH_P5_TO_P9")

def ReadLocalPortFromFile():
    return settings.get("COMMON", "PORT")


def ReadLocalBaudRateFromFile():
    return settings.get("COMMON", "PORT_BAUD_RATE")

def main():
    try:
        settings.read(CONFIG_FILE)

    except:
        print("error openning config.ini")


if __name__ == '__main__':
    main()
```

12.2.10 send_data_to_telegraf.py

```
# Sergio Rueda Teruel. 2018
# Este software ha sido desarrollado para el trabajo fin de máster de la titulación
```

```
# Máster Universitario en Ingeniería de Telecomunicación UOC-URL de la
# Universidad Oberta de Catalunya y lleva por título
# "Diseño de una WSN para la estimación del seeing de la cúpula D080,
# en el Observatorio Astrofísico de Javalambre."
# Este código está sometido a licencia de Reconocimiento-NoComercial-CompartirIgual
# 3.0 España de Creative Commons.

# Este módulo incorpora una serie de funciones que son llamadas por otros módulos para
# el envío de las métricas a telegraf y así este agente las incorporará en la BD
# Se ha hecho uso de la librería pytelegraf


from telegraf.client import TelegrafClient
import time

#DB_ADDR='192.168.1.44'
DB_ADDR='10.10.64.109'
DB_PORT=8094

# Envío de la metrica de fwhm con timestamp de la adquisición
def send_fwhm_with_timestamp(fwhm, time):
    client = TelegrafClient(host=DB_ADDR, port=DB_PORT, tags={'host': 'CPD'})
    client.metric('FWHM', {'C080': fwhm},timestamp=int(time*1000000000))


# Envío de la metrica de fwhm el timestamp es el del momento de la inserción
def send_fwhm(fwhm):
    client = TelegrafClient(host=DB_ADDR, port=DB_PORT, tags={'host': 'CPD'})
    client.metric('FWHM', {'C080': fwhm})

# Envío de la metrica de temperatura de los 4 sensores de un nodo y de su valor de
alimentación
def main(nodo,t1,t2,t3,t4,vcc):
    client = TelegrafClient(host=DB_ADDR, port=DB_PORT, tags={'host':nodo})
    client.metric('Temperatura',{'Sensor1':round(t1,3)})
    client.metric('Temperatura',{'Sensor2':round(t2,3)})
    client.metric('Temperatura',{'Sensor3':round(t3,3)})
    client.metric('Temperatura',{'Sensor4':round(t4,3)})
    client.metric('Temperatura',{'diff_s1-s2':round(t1-t2,3)})
    client.metric('Temperatura',{'diff_s3-s4': round(t3 - t4, 3)})
    client.metric('Temperatura', {'mean': round((t1+t2+t3+t4)/4, 3)})
    client.metric('Power_Supply',{'Power': round(vcc/1000,2)}) # en voltios
    time.sleep(1)


if __name__ == '__main__':
    main('Remoto_1',0,0,0,0,0)
```

12.2.11 write_local_params_xbee.py

```
# Sergio Rueda Teruel. 2018
# Este software ha sido desarrollado para el trabajo fin de master de la titulación
# Máster Universitario en Ingeniería de Telecomunicación UOC-URL de la
# Universidad Oberta de Catalunya y lleva por título
# "Diseño de una WSN para la estimación del seeing de la cúpula D080,
# en el Observatorio Astrofísico de Javalambre."
# Para la realización de este código se han utilizado las librerías Python
# que la empresa Digi (Digi International Inc.) proporciona en su página web
# (https://www.digi.com/blog/xbee/introducing-the-official-digi-xbee-python-library/)
# este código está sometido a licencia de Reconocimiento-NoComercial-CompartirIgual
# 3.0 España de Creative Commons.

# Este módulo se utiliza para la escritura de los parámetros de configuración a través del
# puerto serie del nodo local.


from digi.xbee.devices import XBeeDevice
from digi.xbee.util import utils
import read_sys_config
import read_node_config_file
import sys
sys.tracebacklimit = 0

# Parámetros de conexión con el puerto serie al dispositivo local
PORT=read_sys_config.ReadLocalPortFromFile()
```

```
BAUD_RATE = read_sys_config.ReadLocalBaudRateFromFile()
node_address="local_node"


def main(NodeAddress):
    print(" +-----------------------------------------------+")
    print(" |           Write Local XBee parameters         |")
    print(" +-----------------------------------------------+\n")



    local_device = XBeeDevice(PORT, BAUD_RATE)


    try:
        local_device.open()
       # Get Hardware Models with extended DIO (P5 to P9)
        Hardware_Extended = read_sys_config.ReadHardwareVersionWhithP5ToP9PinsFromFile()
        HV = utils.hex_to_string(local_device.get_parameter("HV"))
       # Set filne name source of the params
        read_node_config_file.set_path_to_file(NodeAddress)

        # Networking

        local_device.set_parameter("ID",
utils.hex_string_to_bytes(read_node_config_file.ReadPanIDFromFile(NodeAddress)))

        local_device.set_parameter("SC",
utils.hex_string_to_bytes(read_node_config_file.ReadScanChannelsFromFile(NodeAddress)))
        local_device.set_parameter("SD",
utils.hex_string_to_bytes(read_node_config_file.ReadScanDurationFromFile(NodeAddress)))
        local_device.set_parameter("ZS",
utils.hex_string_to_bytes(read_node_config_file.ReadZigBeeStackProfileFromFile(NodeAddress)))
        local_device.set_parameter("NJ",
utils.hex_string_to_bytes(read_node_config_file.ReadNodeJoinTimeFromFile(NodeAddress)))
        local_device.set_parameter("NW",
utils.hex_string_to_bytes(read_node_config_file.ReadNetworkWatchdogTimeoutFromFile(NodeAddress
)))
        local_device.set_parameter("JV",
utils.hex_string_to_bytes(read_node_config_file.ReadChannelVerificationFromFile(NodeAddress)))
        local_device.set_parameter("JN",
utils.hex_string_to_bytes(read_node_config_file.ReadJoinNotificationFromFile(NodeAddress)))
        local_device.set_parameter("CE",
utils.hex_string_to_bytes(read_node_config_file.ReadCoordinatorEnableFromFile(NodeAddress)))
        local_device.set_parameter("DO",
utils.hex_string_to_bytes(read_node_config_file.ReadDeviceOptionsFromFile(NodeAddress)))
        local_device.set_parameter("DC",
utils.hex_string_to_bytes(read_node_config_file.ReadDeviceControlsFromFile(NodeAddress)))
        # Addressing
        local_device.set_parameter("DH",
utils.hex_string_to_bytes(read_node_config_file.ReadDestinationAddressHighFromFile(NodeAddress
)))
        local_device.set_parameter("DL",
utils.hex_string_to_bytes(read_node_config_file.ReadDestinationAddressLowFromFile(NodeAddress)
))
        local_device.set_parameter("NI",
bytearray(read_node_config_file.ReadNodeIdentifierFromFile(NodeAddress), 'utf8'))
        local_device.set_parameter("NH",
utils.hex_string_to_bytes(read_node_config_file.ReadMaximumHopsFromFile(NodeAddress)))
        local_device.set_parameter("BH",
utils.hex_string_to_bytes(read_node_config_file.ReadBroadcastRadiusFromFile(NodeAddress)))
        local_device.set_parameter("AR",
utils.hex_string_to_bytes(read_node_config_file.ReadManyToOneRouteBroadcastTimeFromFile(NodeAd
dress)))
        local_device.set_parameter("DD",
utils.hex_string_to_bytes(read_node_config_file.ReadDeviceTypeIdentifierFromFile(NodeAddress))
)
        local_device.set_parameter("NT",
utils.hex_string_to_bytes(read_node_config_file.ReadNodeDiscoveryBackoffFromFile(NodeAddress))
)
        local_device.set_parameter("NO",
utils.hex_string_to_bytes(read_node_config_file.ReadNodeDiscoveryOptionsFromFile(NodeAddress))
)
        local_device.set_parameter("CR",
utils.hex_string_to_bytes(read_node_config_file.ReadPanConflictThresholdFromFile(NodeAddress))
)
        # ZigBee Addressing
```

```
        local_device.set_parameter("SE",
utils.hex_string_to_bytes(read_node_config_file.ReadZigBeeSourceEndPointFromFile(NodeAddress))
)
        local_device.set_parameter("DE",
utils.hex_string_to_bytes(read_node_config_file.ReadZigBeeDestinationEndpointFromFile(NodeAddr
ess)))
        local_device.set_parameter("CI",
utils.hex_string_to_bytes(read_node_config_file.ReadZigBeeClusterIDFromFile(NodeAddress)))
        local_device.set_parameter("TO",
utils.hex_string_to_bytes(read_node_config_file.ReadTransmitOptionsFromFile(NodeAddress)))
        # RF Interfacing
        local_device.set_parameter("PL",
utils.hex_string_to_bytes(read_node_config_file.ReadTxPowerLevelFromFile(NodeAddress)))
        local_device.set_parameter("PM",
utils.hex_string_to_bytes(read_node_config_file.ReadPowerModeFromFile(NodeAddress)))
        # Security
        local_device.set_parameter("EE",
utils.hex_string_to_bytes(read_node_config_file.ReadEncryptionEnableFromFile(NodeAddress)))
        local_device.set_parameter("EO",
utils.hex_string_to_bytes(read_node_config_file.ReadEncryptionOptionsFromFile(NodeAddress)))
        local_device.set_parameter("KY",
utils.hex_string_to_bytes(read_node_config_file.ReadEncryptionKeyFromFile(NodeAddress)))
        local_device.set_parameter("NK",
utils.hex_string_to_bytes(read_node_config_file.ReadNetworkEncryptionKeyFromFile(NodeAddress))
)
        # Serial Interfacing
        local_device.set_parameter("BD",
utils.hex_string_to_bytes(read_node_config_file.ReadBaudRateFromFile(NodeAddress)))
        local_device.set_parameter("NB",
utils.hex_string_to_bytes(read_node_config_file.ReadParityFromFile(NodeAddress)))
        local_device.set_parameter("SB",
utils.hex_string_to_bytes(read_node_config_file.ReadStopBitsFromFile(NodeAddress)))
        local_device.set_parameter("RO",
utils.hex_string_to_bytes(read_node_config_file.ReadPacketizationTimeoutFromFile(NodeAddress))
)
        local_device.set_parameter("D6",
utils.hex_string_to_bytes(read_node_config_file.ReadDIO6ConfigurationFromFile(NodeAddress)))
        local_device.set_parameter("D7",
utils.hex_string_to_bytes(read_node_config_file.ReadDIO7ConfigurationFromFile(NodeAddress)))
        local_device.set_parameter("AP",
utils.hex_string_to_bytes(read_node_config_file.ReadApiEnableFromFile(NodeAddress)))
        local_device.set_parameter("AO",
utils.hex_string_to_bytes(read_node_config_file.ReadApiOutputModeFromFile(NodeAddress)))
        # AT Command Options
        local_device.set_parameter("CT",
utils.hex_string_to_bytes(read_node_config_file.ReadATCommandModeTimeoutFromFile(NodeAddress))
)
        local_device.set_parameter("GT",
utils.hex_string_to_bytes(read_node_config_file.ReadGuardTimesFromFile(NodeAddress)))
        local_device.set_parameter("CC",
utils.hex_string_to_bytes(read_node_config_file.ReadCommandSequenceCharacterFromFile(NodeAddre
ss)))
        # Sleep Modes
        local_device.set_parameter("SP",
utils.hex_string_to_bytes(read_node_config_file.ReadCyclicSleepPeriodFromFile(NodeAddress)))
        local_device.set_parameter("SN",
utils.hex_string_to_bytes(read_node_config_file.ReadNumberOfCyclicSleepPeriodsFromFile(NodeAdd
ress)))
        local_device.set_parameter("SM",
utils.hex_string_to_bytes(read_node_config_file.ReadSleepModeFromFile(NodeAddress)))
        local_device.set_parameter("ST",
utils.hex_string_to_bytes(read_node_config_file.ReadTimeBeforeSleepFromFile(NodeAddress)))
        local_device.set_parameter("SO",
utils.hex_string_to_bytes(read_node_config_file.ReadSleepOptionsFromFile(NodeAddress)))
        local_device.set_parameter("WH",
utils.hex_string_to_bytes(read_node_config_file.ReadWakeHostFromFile(NodeAddress)))
        local_device.set_parameter("PO",
utils.hex_string_to_bytes(read_node_config_file.ReadPollRateFromFile(NodeAddress)))
        # I/O Settings
        local_device.set_parameter("D0",
utils.hex_string_to_bytes(read_node_config_file.ReadDIO0AD0ConfigurationFromFile(NodeAddress))
)
        local_device.set_parameter("D1",
utils.hex_string_to_bytes(read_node_config_file.ReadDIO1AD1ConfigurationFromFile(NodeAddress))
)
        local_device.set_parameter("D2",
utils.hex_string_to_bytes(read_node_config_file.ReadDIO2AD2ConfigurationFromFile(NodeAddress))
```

```
)
        local_device.set_parameter("D3",
utils.hex_string_to_bytes(read_node_config_file.ReadDIO3AD3ConfigurationFromFile(NodeAddress))
)
        local_device.set_parameter("D4",
utils.hex_string_to_bytes(read_node_config_file.ReadDIO4ConfigurationFromFile(NodeAddress)))
        local_device.set_parameter("D5",
utils.hex_string_to_bytes(read_node_config_file.ReadDIO5ConfigurationFromFile(NodeAddress)))
        local_device.set_parameter("D8",
utils.hex_string_to_bytes(read_node_config_file.ReadDIO8ConfigurationFromFile(NodeAddress)))
        local_device.set_parameter("D9",
utils.hex_string_to_bytes(read_node_config_file.ReadDIO9ConfigurationFromFile(NodeAddress)))
        local_device.set_parameter("P0",
utils.hex_string_to_bytes(read_node_config_file.ReadDIO10ConfigurationFromFile(NodeAddress)))
        local_device.set_parameter("P1",
utils.hex_string_to_bytes(read_node_config_file.ReadDIO11ConfigurationFromFile(NodeAddress)))
        local_device.set_parameter("P2",
utils.hex_string_to_bytes(read_node_config_file.ReadDIO12ConfigurationFromFile(NodeAddress)))
        local_device.set_parameter("P3",
utils.hex_string_to_bytes(read_node_config_file.ReadDIO13ConfigurationFromFile(NodeAddress)))
        local_device.set_parameter("P4",
utils.hex_string_to_bytes(read_node_config_file.ReadDIO14ConfigurationFromFile(NodeAddress)))
        if HV == Hardware_Extended: # Not all hardware have this inputs
            local_device.set_parameter("P5",
utils.hex_string_to_bytes(read_node_config_file.ReadDIO15ConfigurationFromFile(NodeAddress)))
            local_device.set_parameter("P6",
utils.hex_string_to_bytes(read_node_config_file.ReadDIO16ConfigurationFromFile(NodeAddress)))
            local_device.set_parameter("P7",
utils.hex_string_to_bytes(read_node_config_file.ReadDIO17ConfigurationFromFile(NodeAddress)))
            local_device.set_parameter("P8",
utils.hex_string_to_bytes(read_node_config_file.ReadDIO18ConfigurationFromFile(NodeAddress)))
            local_device.set_parameter("P9",
utils.hex_string_to_bytes(read_node_config_file.ReadDIO19ConfigurationFromFile(NodeAddress)))
        local_device.set_parameter("PR",
utils.hex_string_to_bytes(read_node_config_file.ReadPullUpResistorEnableFromFile(NodeAddress))
)
        local_device.set_parameter("PD",
utils.hex_string_to_bytes(read_node_config_file.ReadPullUpDownDirectionFromFile(NodeAddress)))
        local_device.set_parameter("LT",
utils.hex_string_to_bytes(read_node_config_file.ReadAssociatedLedBlinkTimeFromFile(NodeAddress
)))
        local_device.set_parameter("RP",
utils.hex_string_to_bytes(read_node_config_file.ReadRssiPwmTimerFromFile(NodeAddress)))
        # I/O Sampling
        local_device.set_parameter("IR",
utils.hex_string_to_bytes(read_node_config_file.ReadIOSamplingRateFromFile(NodeAddress)))
        local_device.set_parameter("IC",
utils.hex_string_to_bytes(read_node_config_file.ReadDigitalIOChangeDetectionFromFile(NodeAddre
ss)))
        local_device.set_parameter("V+",
utils.hex_string_to_bytes(read_node_config_file.ReadSupplyVoltageHihgThresholdFromFile(NodeAdd
ress)))

        # Make params permanent
        local_device.write_changes()  # make changes permanet


        #print parameters
        print("  Success!! All Parameters Were Written  ")
        log = "  Success!! All Parameters Were Written  \n\n"

    except:
        log = "  Sorry, an error has happened during writting operation\n"
        if local_device.is_open():
            local_device.close()
        pass


    finally:
        if local_device is not None and local_device.is_open():
            local_device.close()
        else:
            log = "  No local device found\n"

    return log
```

```
if __name__ == '__main__':
    main(node_address)
```

12.2.12 write_remote_params_xbee.py

```
# Sergio Rueda Teruel. 2018
# Este software ha sido desarrollado para el trabajo fin de máster de la titulación
# Máster Universitario en Ingeniería de Telecomunicación UOC-URL de la
# Universidad Oberta de Catalunya y lleva por título
# "Diseño de una WSN para la estimación del seeing de la cúpula D080,
# en el Observatorio Astrofísico de Javalambre."
# Para la realización de este código se han utilizado las librerías Python
# que la empresa Digi (Digi International Inc.) proporciona en su página web
# (https://www.digi.com/blog/xbee/introducing-the-official-digi-xbee-python-library/)
# este código está sometido a licencia de Reconocimiento-NoComercial-CompartirIgual
# 3.0 España de Creative Commons.

# Este módulo se utiliza para la escritura de los parámetros de configuración a través de
# RF de un nodo remoto, dicha escritura se hace a través del nodo local.

from digi.xbee.devices import XBeeDevice
from digi.xbee.devices import RemoteZigBeeDevice
from digi.xbee.util import utils
from digi.xbee.models.address import XBee64BitAddress
import read_sys_config
import read_node_config_file
import sys
sys.tracebacklimit = 0

# Parámetros de conexión con el puerto serie al dispositivo local
PORT=read_sys_config.ReadLocalPortFromFile()
BAUD_RATE = read_sys_config.ReadLocalBaudRateFromFile()

REMOTE_1 = "0013A200416299FE"


def main(Node_Address):

    NodeAddress = XBee64BitAddress.from_hex_string(Node_Address)

    print(" +-------------------------------------------------------+")
    print(" |  Write Remote XBee (" +Node_Address+ ") parameters      |")
    print(" +------------------------------------------------- --+\n")

    local_device = XBeeDevice(PORT, BAUD_RATE)
    local_device.open()
    remote_device = RemoteZigBeeDevice(local_device, NodeAddress)

    try:


        # Get Hardware Models with extended DIO (P5 to P9)
        Hardware_Extended = read_sys_config.ReadHardwareVersionWhithP5ToP9PinsFromFile()
        HV = utils.hex_to_string(remote_device.get_parameter("HV"))
        # Set file name source of the params
        print("0013A200416299FE")
        read_node_config_file.set_path_to_file(Node_Address)
        #No se puede modificar la PAN ID vía RF porque se pierde conexión

        print(read_node_config_file.ReadScanChannelsFromFile(Node_Address))
        remote_device.set_parameter("SC",
utils.hex_string_to_bytes(read_node_config_file.ReadScanChannelsFromFile(Node_Address)))
        remote_device.set_parameter("SD",
utils.hex_string_to_bytes(read_node_config_file.ReadScanDurationFromFile(Node_Address)))
        remote_device.set_parameter("ZS",
utils.hex_string_to_bytes(read_node_config_file.ReadZigBeeStackProfileFromFile(Node_Address)))
        remote_device.set_parameter("NJ",
utils.hex_string_to_bytes(read_node_config_file.ReadNodeJoinTimeFromFile(Node_Address)))
        remote_device.set_parameter("NW",
utils.hex_string_to_bytes(read_node_config_file.ReadNetworkWatchdogTimeoutFromFile(Node_Addres
s)))
        remote_device.set_parameter("JV",
utils.hex_string_to_bytes(read_node_config_file.ReadChannelVerificationFromFile(Node_Address))
)
        remote_device.set_parameter("JN",
```

```
utils.hex_string_to_bytes(read_node_config_file.ReadJoinNotificationFromFile(Node_Address)))
        remote_device.set_parameter("CE",
utils.hex_string_to_bytes(read_node_config_file.ReadCoordinatorEnableFromFile(Node_Address)))
        remote_device.set_parameter("DO",
utils.hex_string_to_bytes(read_node_config_file.ReadDeviceOptionsFromFile(Node_Address)))
        remote_device.set_parameter("DC",
utils.hex_string_to_bytes(read_node_config_file.ReadDeviceControlsFromFile(Node_Address)))
        # Addressing
        remote_device.set_parameter("DH",
utils.hex_string_to_bytes(read_node_config_file.ReadDestinationAddressHighFromFile(Node_Addres
s)))
        remote_device.set_parameter("DL",
utils.hex_string_to_bytes(read_node_config_file.ReadDestinationAddressLowFromFile(Node_Address
)))
        remote_device.set_parameter("NI",
bytearray(read_node_config_file.ReadNodeIdentifierFromFile(Node_Address), 'utf8'))
        remote_device.set_parameter("NH",
utils.hex_string_to_bytes(read_node_config_file.ReadMaximumHopsFromFile(Node_Address)))
        remote_device.set_parameter("BH",
utils.hex_string_to_bytes(read_node_config_file.ReadBroadcastRadiusFromFile(Node_Address)))
        remote_device.set_parameter("AR",
utils.hex_string_to_bytes(read_node_config_file.ReadManyToOneRouteBroadcastTimeFromFile(Node_A
ddress)))
        remote_device.set_parameter("DD",
utils.hex_string_to_bytes(read_node_config_file.ReadDeviceTypeIdentifierFromFile(Node_Address)
))
        remote_device.set_parameter("NT",
utils.hex_string_to_bytes(read_node_config_file.ReadNodeDiscoveryBackoffFromFile(Node_Address)
))
        remote_device.set_parameter("NO",
utils.hex_string_to_bytes(read_node_config_file.ReadNodeDiscoveryOptionsFromFile(Node_Address)
))
        remote_device.set_parameter("CR",
utils.hex_string_to_bytes(read_node_config_file.ReadPanConflictThresholdFromFile(Node_Address)
))
        # ZigBee Addressing
        remote_device.set_parameter("SE",
utils.hex_string_to_bytes(read_node_config_file.ReadZigBeeSourceEndPointFromFile(Node_Address)
))
        remote_device.set_parameter("DE",
utils.hex_string_to_bytes(read_node_config_file.ReadZigBeeDestinationEndpointFromFile(Node_Add
ress)))
        remote_device.set_parameter("CI",
utils.hex_string_to_bytes(read_node_config_file.ReadZigBeeClusterIDFromFile(Node_Address)))
        remote_device.set_parameter("TO",
utils.hex_string_to_bytes(read_node_config_file.ReadTransmitOptionsFromFile(Node_Address)))
        # RF Interfacing
        remote_device.set_parameter("PL",
utils.hex_string_to_bytes(read_node_config_file.ReadTxPowerLevelFromFile(Node_Address)))
        remote_device.set_parameter("PM",
utils.hex_string_to_bytes(read_node_config_file.ReadPowerModeFromFile(Node_Address)))
        # Security
        remote_device.set_parameter("EE",
utils.hex_string_to_bytes(read_node_config_file.ReadEncryptionEnableFromFile(Node_Address)))
        remote_device.set_parameter("EO",
utils.hex_string_to_bytes(read_node_config_file.ReadEncryptionOptionsFromFile(Node_Address)))
        # No se pueden cambiar las keys vía RF

        # Serial Interfacing
        remote_device.set_parameter("BD",
utils.hex_string_to_bytes(read_node_config_file.ReadBaudRateFromFile(Node_Address)))
        remote_device.set_parameter("NB",
utils.hex_string_to_bytes(read_node_config_file.ReadParityFromFile(Node_Address)))
        remote_device.set_parameter("SB",
utils.hex_string_to_bytes(read_node_config_file.ReadStopBitsFromFile(Node_Address)))
        remote_device.set_parameter("RO",
utils.hex_string_to_bytes(read_node_config_file.ReadPacketizationTimeoutFromFile(Node_Address)
))
        remote_device.set_parameter("D6",
utils.hex_string_to_bytes(read_node_config_file.ReadDIO6ConfigurationFromFile(Node_Address)))
        remote_device.set_parameter("D7",
utils.hex_string_to_bytes(read_node_config_file.ReadDIO7ConfigurationFromFile(Node_Address)))
        remote_device.set_parameter("AP",
utils.hex_string_to_bytes(read_node_config_file.ReadApiEnableFromFile(Node_Address)))
        remote_device.set_parameter("AO",
utils.hex_string_to_bytes(read_node_config_file.ReadApiOutputModeFromFile(Node_Address)))
        # AT Command Options
```

```
        remote_device.set_parameter("CT",
utils.hex_string_to_bytes(read_node_config_file.ReadATCommandModeTimeoutFromFile(Node_Address)
))
        remote_device.set_parameter("GT",
utils.hex_string_to_bytes(read_node_config_file.ReadGuardTimesFromFile(Node_Address)))
        remote_device.set_parameter("CC",
utils.hex_string_to_bytes(read_node_config_file.ReadCommandSequenceCharacterFromFile(Node_Addr
ess)))
        # Sleep Modes
        remote_device.set_parameter("SP",
utils.hex_string_to_bytes(read_node_config_file.ReadCyclicSleepPeriodFromFile(Node_Address)))
        remote_device.set_parameter("SN",
utils.hex_string_to_bytes(read_node_config_file.ReadNumberOfCyclicSleepPeriodsFromFile(Node_Ad
dress)))
        remote_device.set_parameter("SM",
utils.hex_string_to_bytes(read_node_config_file.ReadSleepModeFromFile(Node_Address)))
        remote_device.set_parameter("ST",
utils.hex_string_to_bytes(read_node_config_file.ReadTimeBeforeSleepFromFile(Node_Address)))
        remote_device.set_parameter("SO",
utils.hex_string_to_bytes(read_node_config_file.ReadSleepOptionsFromFile(Node_Address)))
        remote_device.set_parameter("WH",
utils.hex_string_to_bytes(read_node_config_file.ReadWakeHostFromFile(Node_Address)))
        remote_device.set_parameter("PO",
utils.hex_string_to_bytes(read_node_config_file.ReadPollRateFromFile(Node_Address)))
        # I/O Settings
        remote_device.set_parameter("D0",
utils.hex_string_to_bytes(read_node_config_file.ReadDIO0AD0ConfigurationFromFile(Node_Address)
))
        remote_device.set_parameter("D1",
utils.hex_string_to_bytes(read_node_config_file.ReadDIO1AD1ConfigurationFromFile(Node_Address)
))
        remote_device.set_parameter("D2",
utils.hex_string_to_bytes(read_node_config_file.ReadDIO2AD2ConfigurationFromFile(Node_Address)
))
        remote_device.set_parameter("D3",
utils.hex_string_to_bytes(read_node_config_file.ReadDIO3AD3ConfigurationFromFile(Node_Address)
))
        remote_device.set_parameter("D4",
utils.hex_string_to_bytes(read_node_config_file.ReadDIO4ConfigurationFromFile(Node_Address)))
        remote_device.set_parameter("D5",
utils.hex_string_to_bytes(read_node_config_file.ReadDIO5ConfigurationFromFile(Node_Address)))
        remote_device.set_parameter("D8",
utils.hex_string_to_bytes(read_node_config_file.ReadDIO8ConfigurationFromFile(Node_Address)))
        remote_device.set_parameter("D9",
utils.hex_string_to_bytes(read_node_config_file.ReadDIO9ConfigurationFromFile(Node_Address)))
        remote_device.set_parameter("P0",
utils.hex_string_to_bytes(read_node_config_file.ReadDIO10ConfigurationFromFile(Node_Address)))
        remote_device.set_parameter("P1",
utils.hex_string_to_bytes(read_node_config_file.ReadDIO11ConfigurationFromFile(Node_Address)))
        remote_device.set_parameter("P2",
utils.hex_string_to_bytes(read_node_config_file.ReadDIO12ConfigurationFromFile(Node_Address)))
        remote_device.set_parameter("P3",
utils.hex_string_to_bytes(read_node_config_file.ReadDIO13ConfigurationFromFile(Node_Address)))
        remote_device.set_parameter("P4",
utils.hex_string_to_bytes(read_node_config_file.ReadDIO14ConfigurationFromFile(Node_Address)))
        if HV == Hardware_Extended:
            remote_device.set_parameter("P5",
utils.hex_string_to_bytes(read_node_config_file.ReadDIO15ConfigurationFromFile(Node_Address)))
            remote_device.set_parameter("P6",
utils.hex_string_to_bytes(read_node_config_file.ReadDIO16ConfigurationFromFile(Node_Address)))
            remote_device.set_parameter("P7",
utils.hex_string_to_bytes(read_node_config_file.ReadDIO17ConfigurationFromFile(Node_Address)))
            remote_device.set_parameter("P8",
utils.hex_string_to_bytes(read_node_config_file.ReadDIO18ConfigurationFromFile(Node_Address)))
            remote_device.set_parameter("P9",
utils.hex_string_to_bytes(read_node_config_file.ReadDIO19ConfigurationFromFile(Node_Address)))
        remote_device.set_parameter("PR",
utils.hex_string_to_bytes(read_node_config_file.ReadPullUpResistorEnableFromFile(Node_Address)
))
        remote_device.set_parameter("PD",
utils.hex_string_to_bytes(read_node_config_file.ReadPullUpDownDirectionFromFile(Node_Address))
)
        remote_device.set_parameter("LT",
utils.hex_string_to_bytes(read_node_config_file.ReadAssociatedLedBlinkTimeFromFile(Node_Addres
s)))
        remote_device.set_parameter("RP",
utils.hex_string_to_bytes(read_node_config_file.ReadRssiPwmTimerFromFile(Node_Address)))
```

```
        # I/O Sampling
        remote_device.set_parameter("IR",
utils.hex_string_to_bytes(read_node_config_file.ReadIOSamplingRateFromFile(Node_Address)))
        remote_device.set_parameter("IC",
utils.hex_string_to_bytes(read_node_config_file.ReadDigitalIOChangeDetectionFromFile(Node_Addr
ess)))
        remote_device.set_parameter("V+",
utils.hex_string_to_bytes(read_node_config_file.ReadSupplyVoltageHihgThresholdFromFile(Node_Ad
dress)))

        # Make params permanent
        remote_device.write_changes()  # make changes permanet"""

        # print parameters
        log ="  Success!! All Parameters Were Written  \n\n"
        print("  Success!! All Parameters Were Written  ")


    except:
        log = "  Sorry, an error has happened during writting operation\n"
        if local_device.is_open():
            local_device.close()
        pass


    finally:
        if local_device is not None and local_device.is_open():
            local_device.close()

    return log


if __name__ == '__main__':
    main(REMOTE_1)
```

12.2.13 Ejemplo archivo configuración nodo remoto.

Los archivos de configuración se encuentran dentro de la carpeta config, la cual estará situada al mismo nivel que el resto de archivos py de ejecución. Este archivo no es más que un listado de los parámetros de un nodo y que será leído por el módulo de configuración de dicho nodo para fijar esos parámetros.

0013A200415BFC59.ini:

```
[0013A200415BFC59]
ID = 2015
SC = 7FFF
SD = 3
ZS = 0
NJ = FF
NW = 0
JV = 0
JN = 0
CE = 0
DO = 0
DC = 0
DH = 0
DL = FFFF
NI = REMOTO_1
NH = 1E
BH = 0
AR = FF
DD = A000
NT = 96
NO = 0
CR = 3
SE = E8
DE = E8
CI = 11
TO = 0
PL = 4
PM = 1
```

```
EE = 0
EO = 0
KY = 0
NK = 0
BD = 3
NB = 0
SB = 0
RO = 3
D6 = 0
D7 = 1
AP = 1
AO = 0
CT = 64
GT = 3E8
CC = 2B
SP = 1F4
SN = 1
SM = 0
ST = 1388
SO = 0
WH = 0
PO = 0
D0 = 1
D1 = 0
D2 = 0
D3 = 0
D4 = 0
D5 = 1
D8 = 1
D9 = 1
P0 = 1
P1 = 0
P2 = 0
P3 = 1
P4 = 1
P5 = 1
P6 = 1
P7 = 1
P8 = 1
P9 = 1
PR = 1FBF
PD = 1FFF
LT = 0
RP = 28
IR = 0
IC = 0
V+ = 0
```

12.2.14 list_of_nodes.ini

Los archivos de configuración se encuentran dentro de la carpeta config, la cual estará situada al mismo nivel que el resto de archivos py de ejecución. En este archivo se indican el nombre (NI) de los nodos a ser encuestados por el programa para la adquisición de datos, de esta forma hacemos el sistema flexible y dónde no todos los nodos de la red deben ser encuestados si no queremos o la inclusión de uno nuevo es tan sencilla como su configuración y su inclusión en esta lista.

```
REMOTO_1
REMOTO_2
REMOTO_3
REMOTO_4
REMOTO_5
REMOTO_6
REMOTO_6
REMOTO_8
REMOTO_9
```

12.2.15 sys_config.ini

Los archivos de configuración se encuentran dentro de la carpeta config, la cual estará situada al mismo nivel que el resto de archivos py de ejecución.

```
# parámetros generales del sistema
# Dependiendo el tipo de chip que utiliza el sensor tiene un pinout u otro, los que se
utilizan en este proyecto son la versión de hardware 2250 y la versión de
# hardware 2E48. La versión 2250 incluye los pines P5 a P9, mientras que la versión 2E48 no
los incluye, en este archivo se lee que versión de hardware tiene esos
# pines mediante la opción HARDWARE_VERSION_WITH_P5_TO_P9, ya que si se intentan leer/escribir
esos parámetros en una vesrión de hardware que no los tiene el
# modulo devuelve error.

[COMMON]
HARDWARE_VERSION_WITH_P5_TO_P9 = 22 50
PORT = COM6
PORT_BAUD_RATE = 9600
```

12.3 Manual básico de instalación

A continuación, se indica, de manera abreviada pero funcional, una instalación básica de todo el software necesario para replicar el sistema. Este manual se divide entre la parte de adquisición basada en Python3 y que ha sido desarrollada específicamente para este TFM y la parte de almacenamiento y monitorización que utiliza software de código abierto y, por tanto, de un propósito más generalista. Sobre esta última parte, simplemente se ha dado una pincelada ya que existen cientos de páginas web en las que nos podemos apoyar para realizar la instalación.
Dado que el despliegue final se ha hecho sobre Ubuntu 16 LTS se explica aquí como replicarlo, no obstante, sería extrapolable a otras distribuciones Linux/Unix y también a Windows 10 mediante el adecuado cambio de los procedimientos de instalación

12.3.1 Software de adquisición (Python3)

Para el correcto funcionamiento del software desarrollado se hace necesaria la instalación de Python 3 [83] y de las siguientes librerías que no vienen incluidas en la distribución ordinaria:

- Digi-XBee
- Pytelegraf
- Pyserial (La experiencia ha demostrado que algunas distribuciones GNU/Linux ya lo incorporan en el paquete estándar).

Aunque se incorporan las librerías al repositorio de código y se pueden descargar e instalar desde el mismo y bastaría simplemente con copiarlas en el directorio dónde tengamos el resto de librerías, se recomienda, por simplicidad, su instalación mediante el software de administración de paquetes para Python3 pip3:

El programa pip3 puede no venir incluido en la distribución de Python3, pero su instalación es muy sencilla, así en Ubuntu 16 LTS se realiza mediante la ejecución de la siguiente instrucción desde un terminal:

- *sudo apt-get update & apt-get install pip3*

Una vez instalado este paquete, la instalación de las librerías se realizará mediante las siguientes instrucciones también desde el terminal:

- *pip3 install digi-xbee*
- *pip3 install pytelegraf*
- *pip3 install pyserial*

Una vez que tanto Python3 como el resto de librerías estén instaladas, no se hace necesaria ninguna acción más para ejecutar los módulos Python. Simplemente hay que respetar la estructura de archivos que se adjunta al repositorio de código en el que los archivos py comparten directorio con la carpeta config en el interior de la cual estarán los archivos de configuración que anteriormente se han mencionado

12.3.2 Software de almacenamiento y monitorización

Para la instalación de todo el software en Ubuntu 16 LTS basta con ejecutar desde un terminal los siguientes comandos:

A. *Instalación y habilitación de los servicios (influxDB, Telegraf y Grafana):*

1. *Actualizar la lista de paquetes:*
   - *sudo apt-get update*
2. *Instalar las aplicaciones*
   - *sudo apt-get install inlfluxdb telegraf grafana*
3. *Activarlas para que se ejecuten en el arranque*
   - *sudo systemctl enable influxdb*
   - *sudo systemctl enable telegraf*
   - *sudo systemctl enable grafana-server*

B. *Configuración de influxDB. Desde el archivo /etc/influxdb/influxdb.conf:*

1. *Abrir el archivo /etc/influxdb/influxdb.conf y modificar la línea para habilitar la autentificación mediante usuario y contraseña:*
   - *# Determines whether HTTP authentication is enabled.*
     auth-enabled = true
2. *Desde un terminal crear la base de datos, añadir un usuario y darle los permisos, en nuestro caso:*
   - *influx*
   - *CREATE DATABASE telegraf*
   - *CREATE USER "TFM" WITH PASSWORD 'TFM' WITH ALL PRIVILEGES*
   - *GRANT ALL ON telegraf TO TFM*
   - *exit*
   - *sudo systemctl restart influxdb*

C. *Configuración de Telegraf. Desde el archivo /etc/telegraf/telegraf.conf:*

1. *Modificar el archivo /etc/telegraf/telegraf.conf para habilitar la entrada de datos desde el socket udp usado por pytelegraf (descomentar línea y añadir la ip)*

- *[[inputs.socket_listener]]*
  service_address = "udp://10.10.64.109:8094"

(en nuestro caso esa es la ip elegida para la raspberry pi dónde irá todo el sistema instalado)

3. *En el mismo archivo, habilitar la conexión con influxdb*
   - *[[outputs.influxdb]]*
     username="TFM"
     password="TFM"
4. *Descomentar la línea:*
   - *database = "telegraf"*
5. *Desde un terminal reiniciar el servicio:*
   - *sudo systemctl restart telegraf*

D. *Configuración Grafana:*

La configuración de Grafana se hace desde una interface web, basta con poner la dirección IP del servidor en el navegador y el puerto 3000. Por defecto el usuario administrador es:

User: admin
Pass: admin

La conexión con influxDB se realiza por defecto en el puerto 8086 (este puerto de escucha de influxDB es configurable en /etc/influxdb/influxdb.conf)
A continuación, se muestra una captura de la configuración para este proyecto

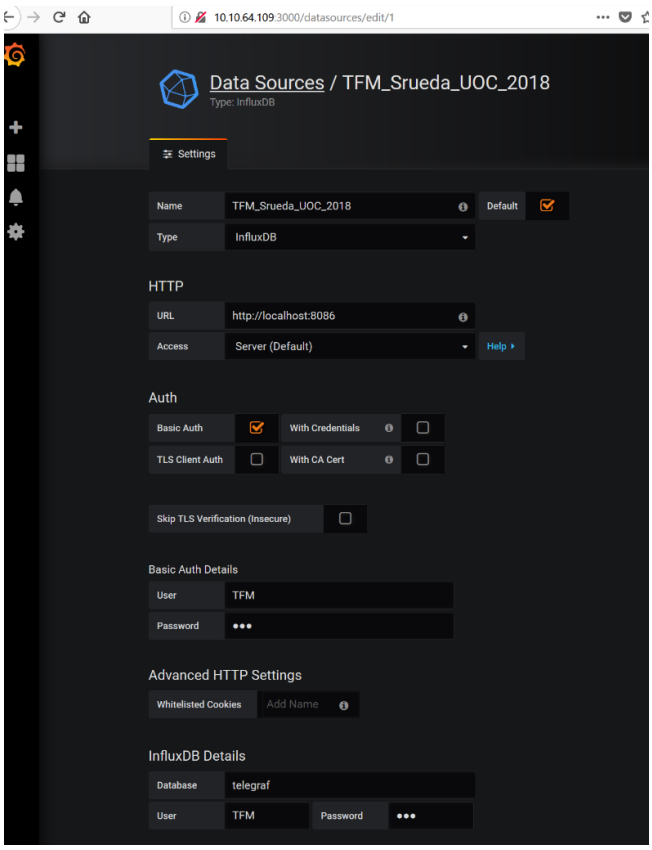


Ilustración 53. Inteface de configuración Grafana. Se indican los datos de conexión para acceder a la base de datos influxdb tal y como se han comentado en los apartados anteriores.

Por otra parte, a la hora de realizar los *dashboard*s para que muestren los datos que hemos capturado, hay que configurar la *query* adecuada dentro de la pantalla de edición de los mismos. El objetivo de estas *queries* es obtener de la base de datos las métricas hemos de mostrar, esto es, por tanto, muy variado y absolutamente dependiente de la aplicación que se desarrolle, no obstante, aquí se muestra un ejemplo de una de nuestras *queries* para mostrar las métricas que recogen las diferencias de temperaturas entre pares de sensores.
Cada *dashboard* puede contener un número ilimitado de paneles (gráficas) y en cada uno de estos paneles podemos mostrar todas las métricas que deseemos simplemente añadiendo *queries* por cada métrica a mostrar.

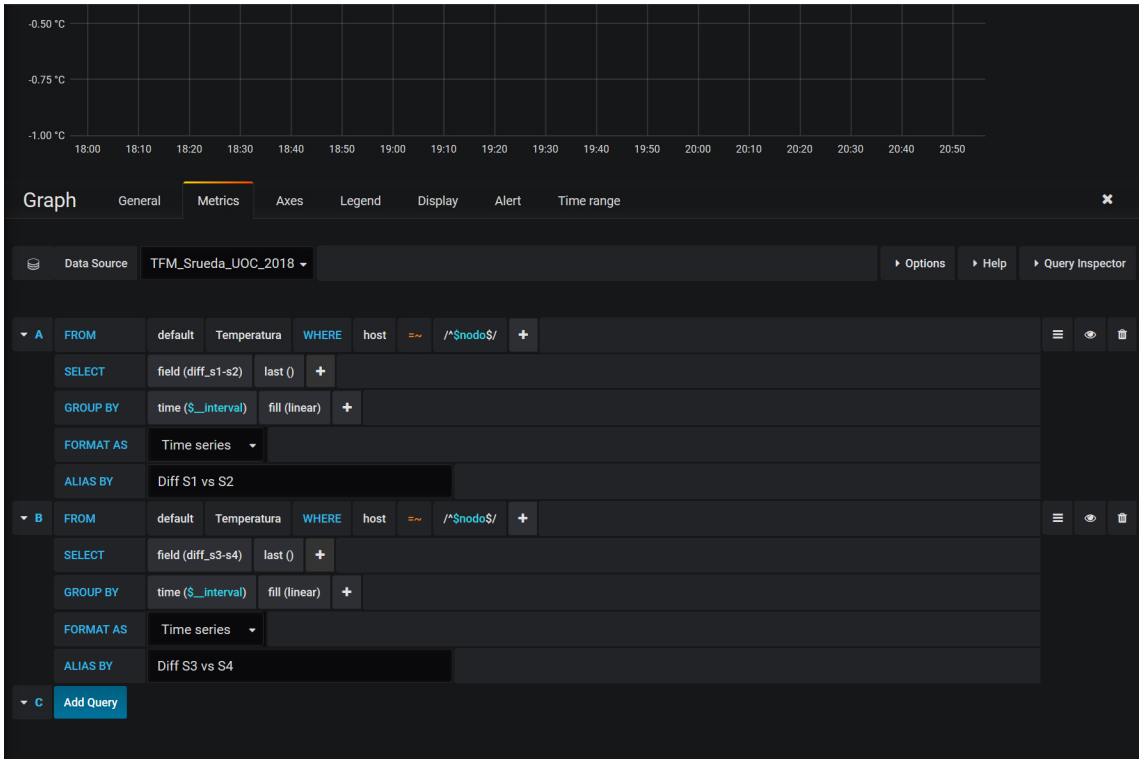


*Ilustración 54. Ejemplo de query Grafana. Vemos que se han realizado dos queries, **A** para la métrica que envía la diferencias entre el sensor 1 y 2 (s1-s2) y la **B** para la métrica que envía la diferencia entre el sensor 3 y sensor 4 (s3-s4). Por otra parte, nótese que en host se usa una variable llamada /^nodo$/, esto nos permite utilizar un único dashboard para todos los nodos, así a esta variable se le asigna el valor "REMOTO_1", "REMOTO_2", etc mediante una pestaña desplegable para que se muestre en el dashboard la información del nodo en cuestión.*

12.4 Manual de usuario

En los siguientes apartados se muestra brevemente, por un lado, como ejecutar los programas Python3 de adquisición y configuración de los módulos XBee, y por otro, como interpretar los datos que se muestran en el sistema de monitorización

12.4.1 Software de adquisición y configuración (Python3)

Para la ejecución del software de adquisición de datos bastará con ejecutar desde un terminal y dentro de la carpeta en la que estén los códigos desarrollados el siguiente comando:

- python3 read_remote_ADC_xbee_multihilo.py &

Tras la ejecución del código irán apareciendo por pantalla mensajes del estatus de todo el proceso.

Para la ejecución del módulo de adquisición de los valores de FWHM basta con ejecutar desde un terminal el siguiente comando (Nótese que el equipo que ejecuta este módulo debe estar en una red válida del observatorio o tener acceso al servidor del centro de cálculo mediante VPN):

- python3 get_fwhm.py&

Al igual que el ejemplo anterior el programa irá mostrando por pantalla los progresos que vaya realizando, es decir, los resultados de las peticiones periódicas a la base de datos del centro de cálculo.

Por último, la ejecución de los módulos de configuración se realiza desde un entorno gráfico, para lanzar este entorno basta con ejecutar desde un terminal el comando:

- python3 gui.py&

Tras él aparecerá la siguiente interface gráfica que se explica a continuación:

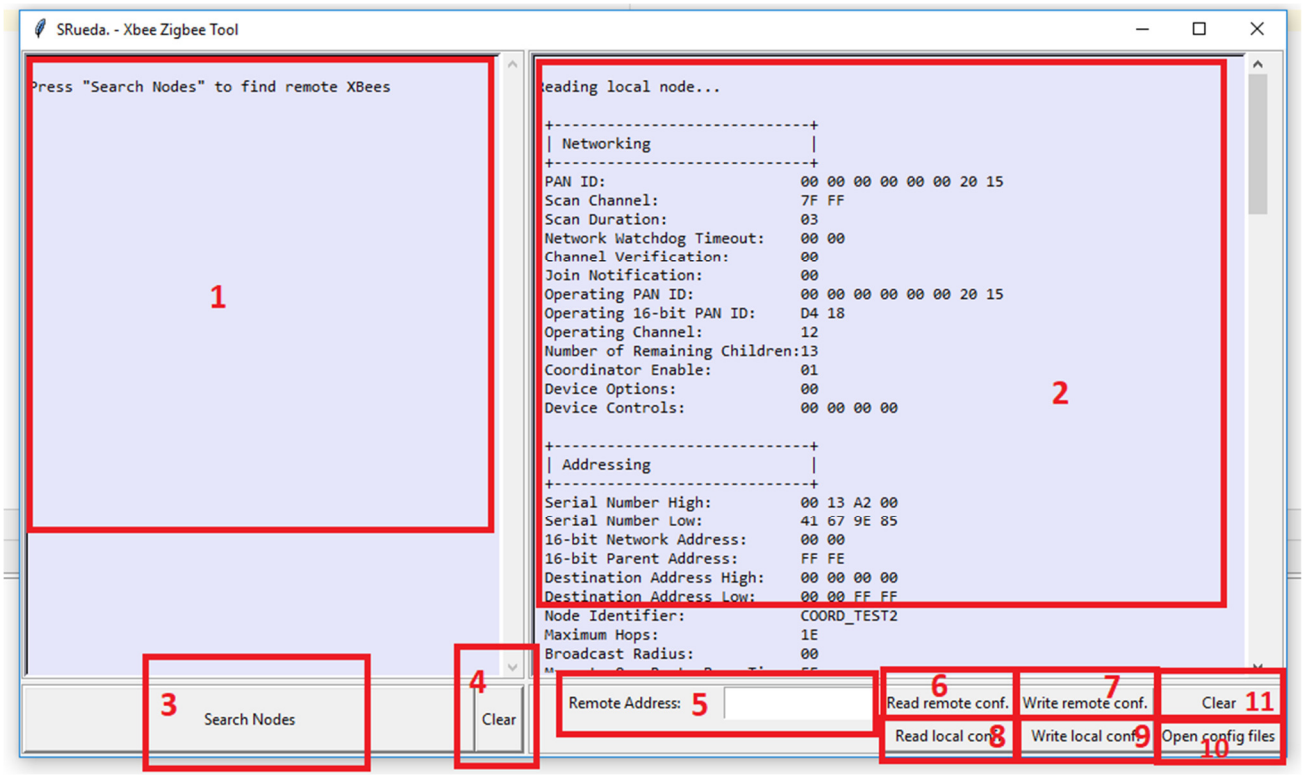


Ilustración 55. Interface gráfica de configuración de los nodos XBee con los botones numerados

Las distintas funcionalidades de esta aplicación gráfica son:

1. Panel de texto donde se mostrarán los nodos que están activos en la red. tras solicitar la búsqueda de los mimos desde el botón 3.
2. Panel de texto donde se mostrarán los parámetros de configuración de los nodos tras pulsar los botones de lectura 6 ó 7, o el éxito o fracaso de una operación de escritura de parámetros a un nodo tras pulsar los botones 7 ó 9.
3. Inicia el procedimiento de búsqueda de nodos activos en la misma red.
4. Limpia los resultados mostrados en el panel 1.

5. Se indica la dirección del nodo remoto cuyos parámetros de configuración queremos leer o escribir.
6. Inicia el procedimiento de lectura de los parámetros de configuración del nodo remoto cuya dirección coincida con la marcada en 5.
7. Ejecuta el procedimiento de escritura de los parámetros de configuración del nodo remoto cuya dirección sea la indicada en 5. Recordemos que dichos parámetros estarán contenidos en un archivo *ini* cuyo nombre debe coincidir con la dirección del nodo.
8. Realiza la lectura de los parámetros del nodo local conectado mediante el puerto serie.
9. Desde este botón se realiza la escritura de los parámetros de configuración del nodo local, dichos parámetros están fijados en el archivo *local_node.ini.*
10. Abre la carpeta donde se almacenan todos los archivos de configuración para su posible edición.
11. Borra los resultados mostrados en el panel 2.

12.4.2 Software de monitorización Grafana

Tras indicar en un navegador web la dirección IP y el puerto adecuados (en nuestra configuración puerto 3000) se presenta el *fronted* Web de monitorización.

1. Desplegable que permite seleccionar un nodo en concreto, desde REMOTO_1 hasta REMOTO_9.
2. En este panel se muestra la diferencia de temperaturas en ºC entre pares de sensores de cada nodo, es decir, *sensor1-sensor2* y *sensor3-sensor4*. Al lado derecho los valores actuales, máximos y mínimos.
3. Aquí se muestran los valores de FWHM en segundos de arco de cada imagen adquirida por el telescopio T080, a la derecha también se muestran los valores máximo, mínimo y actual. Para determinar la contribución en segundos de arco

que las diferencias de temperatura en ºC tienen sobre la FWHM habrá que realizar estudios estadísticos que nos permitan obtener valores de correlación entre ambos parámetros físicos, no obstante, se espera tal y como se ha indicado en la introducción de esta memoria llegar a unas relaciones sencillas como las obtenidas por los trabajos de D. Blanco [11].

4. En esta sección aparecen cuatro gráficas, una por cada termistor del nodo seleccionado, mostrando el histórico de temperaturas en ºC, el valor máximo, mínimo y actual.
5. Se muestra una imagen del nodo seleccionado con el valor de temperatura (ºC) actual calculada para cada sensor.
6. Valor actual del voltaje de alimentación del nodo.
7. Desde aquí se puede elegir el periodo de visualización, permitiéndonos elegir una fecha en concreto, o períodos de tiempo tan amplios que oscilan desde los últimos segundos hasta los últimos años.