



UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Facultat d'Informàtica de Barcelona

FIB

Open Robotic Observatory Control System OpenROCS v2.0

Proyecto Final de Carrera
Ingeniería Técnica en Informática de Sistemas

Alumno: **Josep Sanz Campderrós**
Institut d'Estudis Espacials de Catalunya (IEEC)

Director: **Francesc Vilardell Sallés**
Institut d'Estudis Espacials de Catalunya (IEEC)

Ponente: **Josep Fernandez Ruzafa**
Eng. Sistemes, Automàtica i Inf. Ind. (ESAII)
Universitat Politècnica de Catalunya (UPC)

Mayo de 2015

Agradecimientos



UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH



Quiero agradecer a mi director de proyecto, Francesc Vilardell Sallés, del Institut d'Estudis Espacials de Catalunya (IEEC), la ayuda prestada, las ideas propuestas y sobretodo, los requerimientos que propuso, para llegar al diseño actual de OpenROCS v2.0, así como el trabajo realizado probando el nuevo sistema en el entorno real y ayudando a depurar los problemas que surgieron.

También quiero agradecer a mi profesor de proyecto, Josep Fernandez Ruzafa, del departamento de Eng.Sistemes, Automàtica i Inf.Ind. (ESAII) de la Universitat Politècnica de Catalunya (UPC), la ayuda, consejos y paciencia que ha tenido durante todo este proyecto.

Quiero agradecer a mis compañeros de trabajo, Ignasi Ribas (IEEC-CSIC), Pep Colomé Ferrer (IEEC-CSIC) y a Pere Gil Rodríguez (IEEC), por definir los requerimientos del sistema, supervisar mi trabajo y por la experiencia de usuario.

También, quiero agradecer a todas las personas que han participado en los proyectos del OAdM, del TJO y de SQT, la valiosa labor de conseguir que estos proyectos sean realidad, así como todo el conocimiento aportado.

Como último punto de esta sección, quiero agradecer al Institut d'Estudis Espacials de Catalunya (IEEC) la oportunidad de poder presentar en este PFC mi trabajo realizado durante estos años en este centro de investigación.

*Dedicado a mi mujer Montse
y a mis hijas Itziar y Ainhoa.*

Índice

1. Introducción	17
1.1. ¿Qué es el TJO?	17
1.1.1. Instalaciones	17
1.1.2. Modo de funcionamiento	19
1.1.3. Casos científicos	19
1.2. ¿Qué es el OAdM?	21
1.2.1. Localización	23
1.2.2. Descubriendo la sierra del Montsec	23
1.2.3. Calidad del cielo nocturno	24
1.3. Objetivos	25
1.4. Estructura de la memoria	25
2. Requerimientos científicos	27
2.1. Operaciones del telescopio	27
2.2. Capas de control y flujo de trabajo	28
2.3. Control de flujo de datos global	28
3. Sistema robótico	31
3.1. Telescopio	31
3.2. Cúpula	33
3.3. Instrumentación	34
3.3.1. MEIA	34
3.3.2. ARES	37
3.4. Sistemas de monitorización	39
3.4.1. Dispositivos para la monitorización meteorológica	39
3.4.2. Dispositivos para la monitorización de tiempo	42
3.5. Sistema de control	43
3.5.1. OpenROCS v1.0	43
3.5.2. Software de control TALON	43
3.6. Sistemas auxiliares	48
3.6.1. Alimentación eléctrica	48
3.6.2. Aislamiento eléctrico	50

3.6.3. Internet y LAN	50
4. Requerimientos del nuevo software de control	53
5. Estudio de mercado	55
5.1. RTS2	55
5.2. ASCOM	56
5.3. TALON	57
5.4. INDI	57
5.5. OROCOS	58
5.6. ROS	59
5.7. Conclusiones	60
6. Diseño del nuevo software de control	61
6.1. Diseño de los módulos	62
6.1.1. El servicio Servidor	63
6.1.2. El servicio Broadcast	64
6.1.3. El servicio Monitor	65
6.1.4. El servicio Scheduler	66
6.1.5. La herramienta Cliente	67
6.2. Interfaces del sistema	69
6.2.1. Señales y tuberías (pipes)	69
6.2.2. Sockets TCP/IP	69
6.2.3. Sockets UDP/IP	70
6.2.4. Comandos shell	70
6.3. Configuración del sistema	70
6.3.1. El fichero config.xml	71
6.3.2. El fichero variables.xml	71
6.3.3. El fichero monitor.xml	72
6.3.4. El fichero scheduler.xml	73
6.4. Los nodos de proceso	74
7. Tareas previas a la implementación	75
7.1. Sistema de control del telescopio	75
7.2. Sistemas de meteorología	76

8. Implementación	77
8.1. Lenguaje de programación	77
8.2. Organización del código fuente	77
8.3. Programación del diseño elegido	77
8.4. Funciones heredadas del proyecto SaltOS	78
8.5. Funciones y servicios de OpenROCS	78
8.6. Acciones de OpenROCS	79
8.7. Licencia usada	79
8.8. Ficheros XML usados en el TJO	80
9. Resultados	81
9.1. Patentes	82
9.2. Resultados científicos	82
9.3. Publicaciones técnicas	83
9.4. SuperWASP Qatar Telescope (SQT)	85
10. Planificación y costes	87
10.1. Valoración económica del OAdM y del TJO	87
10.2. Planificación del proyecto	88
10.3. Costes de ejecución	89
11. Conclusiones y trabajo futuro	91
11.1. Conclusiones	91
11.2. Trabajo futuro	91
12. Anexos	93
12.1. El fichero config.xml	93
12.1.1. El nodo <server>	93
12.1.2. El nodo <broadcast>	93
12.1.3. El nodo <debug>	94
12.1.4. El nodo <shell>	94
12.1.5. El nodo <timeout>	94
12.1.6. El nodo <polling>	95
12.1.7. El nodo <retries>	95
12.1.8. El nodo <ini_set>	96

12.1.9. El nodo <putenv>	96
12.2. Los nodos de proceso	96
12.2.1. El nodo <action>	96
12.2.2. El nodo <shell>	97
12.2.3. El nodo <php>	98
12.2.4. El nodo <choose>	99
12.2.5. El nodo <send>	102
12.2.6. El nodo <log>	102
12.3. Sistema de control del telescopio	105
12.3.1. talon_fifo	105
12.3.2. talon_alias	110
12.4. Sistemas de meteorología	112
12.4.1. previstorm.c	112
12.4.2. davis.c	115
12.4.3. emsmc.php	119
12.4.4. EMSMC.xml	121
12.4.5. vaisala.php	121
12.4.6. raindetect.c	123
12.4.7. cloudsensor.c	124
12.4.8. sunpos.py	131
12.4.9. snmpd.conf	132
12.5. Programación del diseño elegido	132
12.5.1. orocs	132
12.6. Funciones heredadas del proyecto SaltOS	133
12.6.1. array2xml.php	133
12.6.2. saltos.php	134
12.6.3. xml2array.php	138
12.7. Funciones y servicios de OpenROCS	139
12.7.1. args.php	139
12.7.2. broadcast.php	139
12.7.3. checks.php	140
12.7.4. childs.php	141
12.7.5. comm.php	144

12.7.6. define.php	145
12.7.7. functions.php	149
12.7.8. gc.php	152
12.7.9. monitor.php	153
12.7.10.pipes.php	154
12.7.11.process.php	155
12.7.12scheduler.php	157
12.7.13server.php	158
12.7.14signals.php	159
12.7.15stack.php	160
12.8. Acciones de OpenROCS	162
12.8.1. crontab.php	162
12.8.2. help.php	162
12.8.3. reload.php	162
12.8.4. restart.php	162
12.8.5. shell.php	162
12.8.6. start.php	162
12.8.7. start0.php	162
12.8.8. start1.php	162
12.8.9. start2.php	163
12.8.10stop.php	163
12.8.11stop1.php	163
12.8.12stop2.php	163
12.9. Licencia usada	163
12.9.1. Cabecera OpenROCS v2.0	163
12.9.2. Cabecera SaltOS 3.1	164
12.10 Ficheros XML usados en el TJO	164
12.10.1.config.xml	164
12.10.2.variables.xml	165
12.10.3.monitor.xml	167
12.10.4scheduler.xml	181
12.10.5tjo.xml	192
12.10.6dome.xml	196

12.10.7.meia.xml	198
12.10.8.meteo.xml	200
12.10.9.telrun.xml	203

13. Bibliografía	207
-------------------------	------------

Índice de figuras

1.	Telescopio Joan Oró	18
2.	Sala de control	18
3.	Sistema de control original	18
4.	Imagen aérea del OAdM	19
5.	Edificio del TJO	19
6.	Plano de la planta 0 del TJO	19
7.	Plano de la planta 1 del TJO	19
8.	M82 y SN2014, Galaxia M82 con la supernova SN2014 indicada con líneas amarillas	20
9.	Arp316, Galaxias en interacción	20
10.	M1, Nebulosa del Cangrejo	21
11.	Cometa Lulin C 2007 N3	21
12.	NGC7331, Galaxia espiral, en la constelación de Pegasus	21
13.	M13, Gran cúmulo globular de Hércules	21
14.	Telescopio Joan Oró	22
15.	Telescopio Fabra-ROA-Montsec	22
16.	Cámara all-sky del IEEC	22
17.	Unidad XO del STScI	22
18.	Estación meteorológica del SMC	22
19.	Estación XPCA	22
20.	Mapa de la ubicación del OAdM	23
21.	Vista panorámica del OAdM	24
22.	Vista panorámica del OAdM	24
23.	Fotografía del TJO	24
24.	Brillo del cielo de la Península Ibérica (créditos en el punto 9 de la bibliografía)	24
25.	Operaciones del telescopio	27
26.	Capas de control y flujo de trabajo	28
27.	Elementos del sistema robótico	31
28.	Vista lateral del TJO	32
29.	Vista trasera del TJO	32
30.	Vista lateral del TJO	32
31.	Vista de la óptica del TJO	32

32. Fotografía del espejo primario del TJO	33
33. Fotografía de la colimación del TJO	33
34. Cúpula del TJO	33
35. Cámara Finger Lakes Instrumentation	35
36. ML4240 Quantum Efficiency	35
37. Rueda de filtros del TJO	35
38. Respuesta de cada filtro	35
39. Lampara para hacer flats de cúpula	36
40. Pantalla para hacer flats de cúpula	36
41. Óptima y VPHs del instrumento ARES	37
42. Sistema completo del instrumento ARES	38
43. VPHs de ARES sin guía	38
44. VPHs sobre la guía	38
45. Cámara Andor DU940P	39
46. DU940P Quantum Efficiency	39
47. Sensor externo previstorm	40
48. Electrónica control previstorm	40
49. Estación meteorológica DAVIS Vantage Pro II	40
50. Estación meteorológica Campbell Scientific del SMC	41
51. Estación meteorológica Vaisala del TFRM	41
52. Sensor de lluvia IRSS88	42
53. Sensor de nubes Boltwood Cloud Sensor II	42
54. Antena GPS (Garmin)	43
55. Interfaz xobs de TALON (Main Window)	45
56. Interfaz xobs de TALON (Software Hand Paddle y Find Homes)	45
57. Interfaz xcamera de TALON	46
58. Interfaz telsched de TALON	47
59. Interfaz XEphem	48
60. Placas solares	49
61. Baterías	49
62. Control de carga de baterías	49
63. Cuadro eléctrico general	49
64. Depósito de combustible	50

65. Generador eléctrico diesel	50
66. Zona superior armario de comunicaciones	51
67. Zona inferior armario de comunicaciones	51
68. Logo del software de control de observatorios RTS2	55
69. Interfaz de usuario de RTS2	56
70. Logo del paquete de software ASCOM	56
71. Logo del software de control de observatorios TALON	57
72. Logo del sistema de control distribuido INDI	57
73. Logo del software de control robótico OROCOS	58
74. Logo del software de control robótico ROS	59
75. Diagrama de bloques básico de OpenROCS v2.0	61
76. Diagrama de bloques completo de OpenROCS v2.0	62
77. Diagrama de bloques del servicio servidor de OpenROCS v2.0	63
78. Diagrama de bloques del servicio broadcast de OpenROCS v2.0	64
79. Diagrama de bloques del servicio monitor de OpenROCS v2.0	65
80. Diagrama de bloques del servicio scheduler de OpenROCS v2.0	66
81. Diagrama de bloques de la herramienta cliente de OpenROCS v2.0	67
82. Herramienta cliente de OpenROCS v2.0	68
83. Noches observables vs observadas por año	81
84. Fracción de noches observadas por año	81
85. Imágenes obtenidas por año	82
86. Horas observadas por año	82
87. Poster presentado en el SPIE de Amsterdam 2012	84
88. Exterior telescopio SuperWASP y SQT	85
89. Exterior telescopio SQT	85
90. Telescopio SQT	85
91. Armario de comunicaciones	85
92. Pantalla de control principal de OpenROCS v2.0 para el telescopio SQT	86
93. Pantalla de registros de OpenROCS v2.0 para el telescopio SQT	86
94. Distribución de la dedicación por tareas	88
95. Diagrama de Gantt de la planificación del proyecto	88
96. Diagrama de Gantt de la ejecución del proyecto	89
97. Interfaz gráfico para el control de una implementación de STRIPS	92

98. Workflow de la función process	155
--	-----

Índice de tablas

1.	Magnitud límite para una relación S/N=100	35
2.	Observaciones desde julio del año 2010 hasta finales del año 2014	81
3.	Inversión en la construcción y equipamiento para el OAdM y el TJO	87
4.	Inversión y mantenimiento entre los años 2004 y 2013 para el OAdM y el TJO	88
5.	Costes de ejecución de este proyecto	89
6.	Resultado del ejemplo de la ejecución del nodo <fromiter>	100
7.	Resultado del ejemplo de la ejecución del nodo <everyiter>	100
8.	Resultado del ejemplo de la ejecución del nodo <fromiter> conjuntamente con <everyiter> . .	101
9.	Resultado del ejemplo de la ejecución del nodo <untiliter>	101

Índice de códigos

1. Ejemplo correcto de uso de CDATA en un nodo XML	70
2. Ejemplo erróneo de uso sin CDATA en un nodo XML	70
3. Ejemplo de formato de un fichero XML	71
4. Formato del fichero variables.xml	71
5. Formato del fichero monitor.xml	72
6. Formato del fichero scheduler.xml	73
7. Formato del nodo <server> del fichero config.xml	93
8. Formato del nodo <broadcast> del fichero config.xml	93
9. Formato del nodo <debug> del fichero config.xml	94
10. Formato del nodo <shell> del fichero config.xml	94
11. Formato del nodo <timeout> del fichero config.xml	95
12. Formato del nodo <polling> del fichero config.xml	95
13. Formato del nodo <retries> del fichero config.xml	95
14. Formato del nodo <ini_set> del fichero config.xml	96
15. Formato del nodo <putenv> del fichero config.xml	96
16. Formato del nodo <action> de los nodos de proceso	96
17. Ejemplo de como ejecutar a una acción externa desde un nodo <action>	97
18. Ejemplo de como ejecutar a una acción externa desde un nodo <action>	97
19. Formato del nodo <shell> de los nodos de proceso	97
20. Formato del nodo <timeout> de los nodos de proceso	98
21. Formato del nodo <ontimeout> de los nodos de proceso	98
22. Formato del nodo <php> de los nodos de proceso	98
23. Ejemplo del formato de una tarea periódica	98
24. Formato del nodo <choose> de los nodos de proceso	99
25. Ejemplo de uso del nodo <eval>	99
26. Ejemplo del resultado del nodo <eval>	100
27. Ejemplo del nodo <eval> de los nodos de proceso	100
28. Formato del nodo <fromiter> de los nodos de proceso	100
29. Formato del nodo <everyiter> de los nodos de proceso	100
30. Formato del nodo <fromiter> usando conjuntamente con <everyiter>	101
31. Formato del nodo <utiliter> de los nodos de proceso	101

32. Formato del nodo <fromset> de los nodos de proceso	101
33. Formato del nodo <send> de los nodos de proceso	102
34. Formato del nodo <log> de los nodos de proceso	102
35. Fragmento del fichero user.log generado por el TJO en una noche de observación	102
36. Contenido del script talon_fifo	105
37. Contenido del fichero talon_alias	110
38. Contenido del fichero previstorm.c	112
39. Contenido del fichero davis.c	115
40. Contenido del fichero emsmc.php	119
41. Contenido del fichero EMSMC.xml	121
42. Contenido del fichero vaisala.php	121
43. Contenido del fichero raindetect.c	123
44. Contenido del fichero cloudsensor.c	124
45. Contenido del fichero sunpos.py	131
46. Contenido del fichero snmpd.conf	132
47. Contenido del fichero orocs de OpenROCS v2.0	132
48. Contenido del fichero define.php de OpenROCS v2.0	146
49. Ejemplo de un posible valor de STDOUT tras ejecutar un nodo <shell>	150
50. Conjunto de valores para STDOUT* tras ejecutar la función make_array_vars	150
51. Cabecera usada por el proyecto OpenROCS v2.0	163
52. Cabecera usada por el proyecto SaltOS	164

1. Introducción

Los telescopios robóticos son instalaciones totalmente autónomas. No requieren de interacción humana para que funcionen, proporcionando un aumento de eficacia a pesar del desarrollo de sistemas más complejos. Tienen que emular la respuesta humana a todos los acontecimientos que perturban el flujo normal y cualquier decisión tiene que ser prevista en la fase del diseño. El funcionamiento defectuoso de cualquier elemento de hardware o software debe ser detectable y se debe de preparar una respuesta, en cada caso, para asegurar la fiabilidad y la seguridad.

Podemos distinguir diferentes niveles de control que dependen de la complejidad de la operación desatendida: ejecución normal del sistema, monitorización del entorno y el nivel de respuesta para las condiciones cambiantes. Los telescopios robóticos han sido históricamente enfocados principalmente en automatizar la ejecución de tareas y los procesos realizados para el control del sistema (es decir, supervisar el entorno para asegurar la seguridad del sistema). Sin embargo, la eficiencia de una instalación astronómica también puede ser aumentada automatizando todos los pasos desde el envío de propuestas hasta la recogida de resultados por parte de los usuarios, y principalmente en la planificación de tareas, en la ejecución de tareas y en el procesado de datos. Esto es, sobre todo, importante para observatorios multiuso y multiusuarios que manejan diferentes tipos de programas y requieren una gestión más inteligente.

El nivel de la respuesta dinámica de un sistema a las condiciones que cambian es definido por el tipo de retro-alimentación que el sistema tiene para cualquiera de los procesos en ejecución y cómo esta retroalimentación es usada automáticamente para reaccionar con acciones correctivas o reactivas. Los observatorios estáticos (o de lazo abierto) son aquellos que, por ejemplo, no reaccionan a condiciones que cambian excepto en aquellas situaciones que existe un riesgo, como malas condiciones climáticas (es decir, que se cierre de la cúpula cuando llueve). Los observatorios dinámicos (o de lazo cerrado), usan la retro-alimentación de forma continuada (ya sea la monitorización del entorno o la calidad de los datos adquiridos) para dirigir acciones, en caso necesario, para restaurar cualquier componente que funciona mal o modificar el horario. Las instalaciones dinámicas están preparadas para reaccionar eficazmente según los acontecimientos (ya sean tareas rutinarias o tareas no rutinarias) proporcionando un alto nivel de eficiencia.

El diseño operacional de la mayoría de los telescopios robóticos especifica un control de flujo de datos simple, principalmente dedicado a la ejecución de tareas basadas en el modelo de control estático. Su autonomía es completamente baja, pero son una solución buena ya que representan un desarrollo más simple. Otros telescopios robóticos usan planificadores y/o rutinas de procesado de datos para completar un control de flujo dinámico, consiguiendo un alto nivel de autonomía: RAPid Telescopes for Optical Response (RAPTOR), the Bradford Robotic Telescope (BRT), the STELLar Activity project (STELLA) or the Liverpool Telescope (LT).

1.1. ¿Qué es el TJO?

El Telescopio Joan Oró (TJO) es un telescopio robótico de la clase de un metro. La construcción de este telescopio robótico motivó el desarrollo del Observatori Astronòmic del Montsec (OAdM), un sitio dedicado a albergar instalaciones astronómicas de investigación. Este es el telescopio más grande de Cataluña (0.8m) llamado Joan Oró en honor al famoso bioquímico catalán, pionero de la astrobiología.

El TJO está operado desde el año 2007 por el Institut d'Estudis Espacials de Catalunya (IEEC).

1.1.1. Instalaciones

El TJO es un telescopio de 0.8 m fabricado por Optical Mechanics Inc. (OMI), está equipado con una cúpula automática de 6.15-m de diámetro fabricada por Baader Planetarium GmbH y con una cámara de adquisición de imágenes fotométricas (MEIA) como instrumento de primera luz.

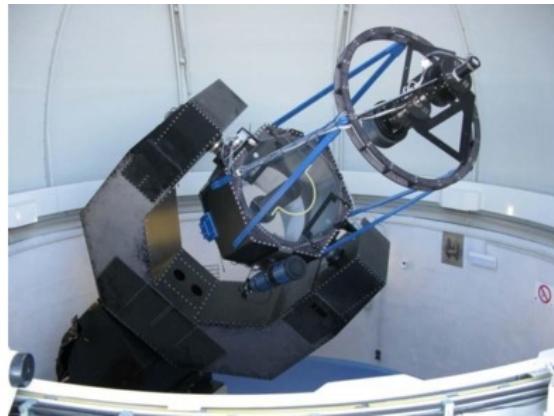


Figura 1: Telescopio Joan Oró

Varios instrumentos monitorizan constantemente el entorno: dos estaciones meteorológicas, una antena GPS, un detector de tormentas, etc. Una conexión de fibra óptica con 100 Mbps de ancho de banda proporciona la comunicación externa necesaria para el acceso remoto.



Figura 2: Sala de control



Figura 3: Sistema de control original

Una arquitectura de software complejo gestiona todas las operaciones del observatorio. Esta arquitectura se gestiona, principalmente con OpenROCS, un software de código abierto desarrollado para controlar observatorios robóticos. El control a bajo nivel del telescopio y de la cúpula se realizan a través del software TALON.



Figura 4: Imagen aérea del OAdM



Figura 5: Edificio del TJO

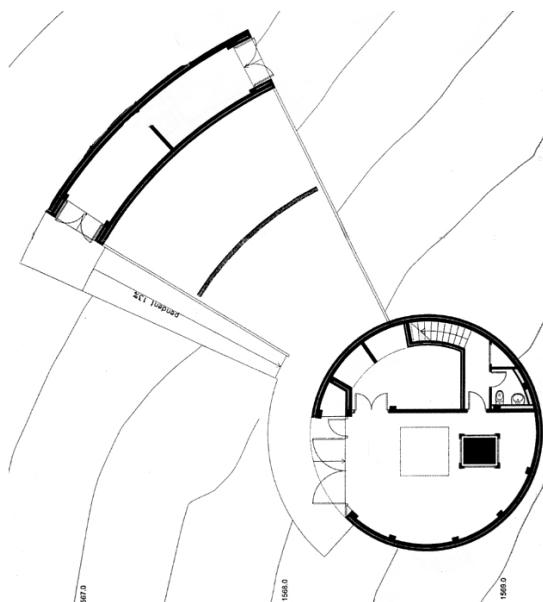


Figura 6: Plano de la planta 0 del TJO

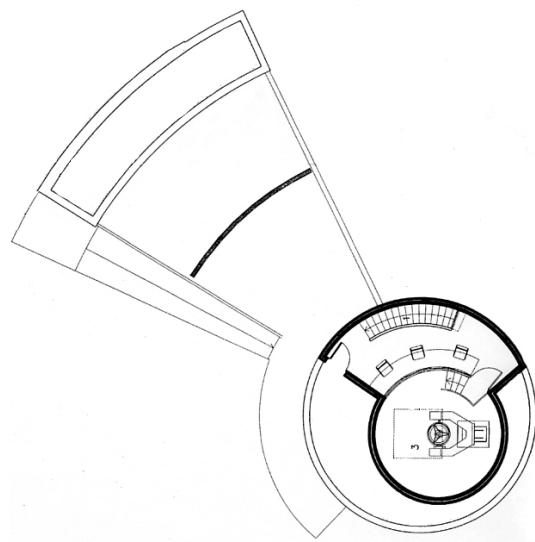


Figura 7: Plano de la planta 1 del TJO

1.1.2. Modo de funcionamiento

Las necesidades de los casos científicos y el aislamiento del lugar, hicieron del funcionamiento robótico un requisito indispensable para el funcionamiento nominal del observatorio. Los observadores sólo tienen que mandar las propuestas con las fuentes de estudio y recuperar los datos y las imágenes adquiridas. El telescopio está siendo preparado para garantizar una alta calidad en las observaciones astronómicas mediante este modo de funcionamiento.

1.1.3. Casos científicos

El TJO es una instalación de propósito general y, como tal, lleva a cabo una variedad de observaciones relacionadas con diversos casos científicos. Dado su tamaño, el principal nicho científico del TJO está en la astronomía de series temporales, donde la alta cadencia y las observaciones continuas son el requisito primordial.



Figura 8: M82 y SN2014, Galaxia M82 con la supernova SN2014 indicada con líneas amarillas

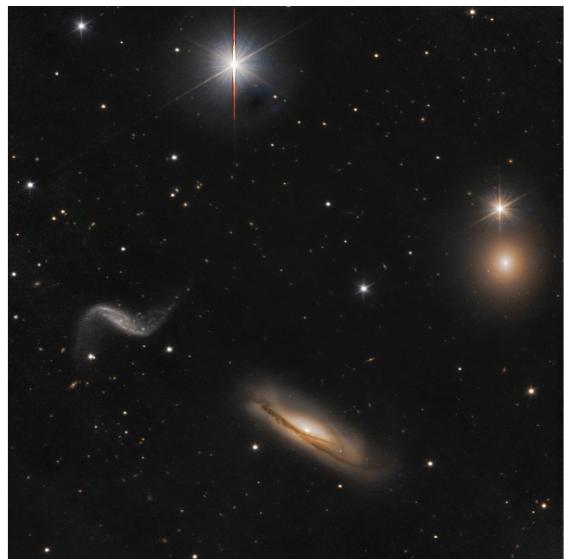


Figura 9: Arp316, Galaxias en interacción

Su principal ventaja es un modo de funcionamiento flexible que permite la observación de las fuentes para períodos de tiempo prolongados y también la posibilidad de un corto tiempo de reacción, potencialmente tan corto como un minuto o menos. Dadas estas características, los posibles casos de ciencia del TJO incluyen:

- Investigación de exoplanetas (con posible seguimiento de planetas en tránsito conocidos o búsquedas específicas de los objetos individuales).
- Binarias eclipsantes (para comprender las propiedades y la estructura estelar).
- Variables pulsantes (para sondear el interior de las estrellas).
- Estrellas variables evolucionadas (gigantes y supergigantes).
- Actividad estelar (para entender la dinamo magnética y para calibrar el tiempo de decaimiento de este tipo de actividad).
- Variabilidad de los núcleos galaxias activas (relacionadas con el proceso de acreción estocástico).
- Objetos del Sistema Solar (seguimiento de asteroides, objetos cercanos a la Tierra, cometas).
- Supernovas (con el valor añadido de obtener fotometría temprana).
- Binarias de rayos X (variabilidad debida a la rotación y a los fenómenos de acreción).
- Novas (también con posibles datos tempranos).
- Contrapartidas ópticas de estallidos de rayos gamma (GRBs).
- Cualquier fenómeno transitorio en general.

Los casos de ciencia anteriores requieren una importante flexibilidad en la programación de la noche, necesitando sistemas que reaccionen rápidamente a las alertas de observación relacionadas con los GRBs, las supernovas y fenómenos similares de tiempo crítico. La participación en las redes de observatorios robóticos permitiría llevar a cabo, por ejemplo, observaciones que requieren una cobertura continua en el tiempo.



Figura 10: M1, Nebulosa del Cangrejo

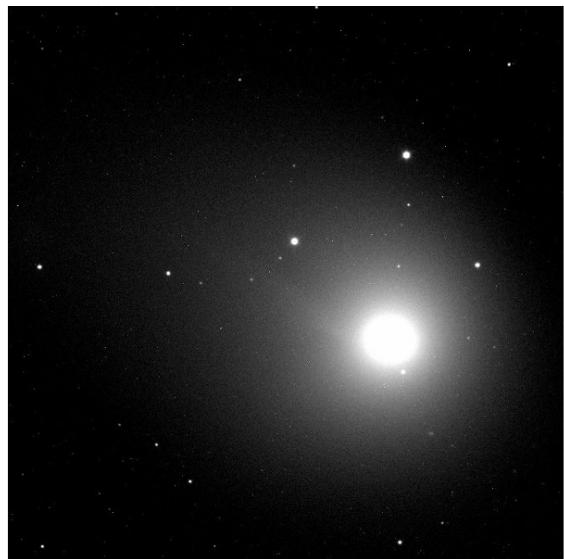


Figura 11: Cometa Lulin C 2007 N3

De manera similar, el TJO puede utilizarse como una instalación de apoyo en ciertas misiones espaciales para recoger datos fotométricos y astrométricos. El TJO ofrece tiempo a la comunidad astronómica a través de propuestas competitivas revisadas mediante un Comité de Asignación de Tiempo (CAT) independiente.



Figura 12: NGC7331, Galaxia espiral, en la constelación de Pegaso



Figura 13: M13, Gran cúmulo globular de Hércules

1.2. ¿Qué es el OAdM?

El proyecto del Observatori Astronòmic del Montsec (OAdM) es apoyado por la Generalitat de Catalunya.

Varias instalaciones de investigación se han instalado en el OAdM: el telescopio TJO (Telescopio Joan Oró), propiedad de la Generalitat de Catalunya y instalado y gestionado por el Institut d'Estudis Espacials de

Catalunya (IEEC); el telescopio TFRM (Telescopi Fabra-ROA-Montsec) de la Real Academia de Ciencias y Artes de Barcelona y el Real Observatorio de la Armada; una cámara all-sky para la detección de meteoros y bólidos; una unidad XO para el descubrimiento de exoplanetas del Space Telescope Science Institute; una estación para el monitoreo de partículas contaminantes en el aire del gobierno catalán (XVPCA, Departament de Territori i Sostenibilitat) gestionada por el Institut de Diagnosi Ambiental i Estudis de l'Aigua; y una estación meteorológica automática del Servei Meteorològic de Catalunya (SMC) del gobierno catalán.



Varias universidades e institutos de investigación han contribuido activamente con la supervisión científico-técnica y mano de obra para el desarrollo del proyecto OAdM: el Institut d'Estudis Espacials de Catalunya (IEEC), la Universitat de Barcelona (UB), la Universitat Politècnica de Catalunya (UPC), el Consejo Superior de Investigaciones Científicas (CSIC) y la Fundació Joan Oró (FJO).



Figura 14: Telescopio Joan Oró



Figura 15: Telescopio Fabra-ROA-Montsec



Figura 16: Cámara all-sky del IEEC



Figura 17: Unidad XO del STScI



Figura 18: Estación meteorológica del SMC



Figura 19: Estación XVPCA

1.2.1. Localización

El OAdM se encuentra a una altitud de 1570m en la sierra del Montsec, a 50km al sud de los Pirineos centrales y 50km al norte de la ciudad de Lleida (Cataluña, España).

Coordenadas geográficas del OAdM:

- Longitud: 00°43'46" E
- Latitud: 42°03'05" N
- Altura: 1570m



Figura 20: Mapa de la ubicación del OAdM

1.2.2. Descubriendo la sierra del Montsec

La zona del Montsec ofrece un rico patrimonio natural i cultural. Se incluye en el plan de espacios de interés natural (PEIN) definido por gobierno catalán (Generalitat de Catalunya). Estas áreas naturales disfrutan de una protección especial para preservar su alto valor ecológico y paisajístico. La variedad de espacios naturales en la sierra del Montsec lo hacen ideal para deportes al aire libre y actividades de ocio en contacto con la naturaleza, aprovechando los elementos y recursos de las montañas, los valles, los ríos y los lagos. Poneros en contacto con el Parc Astronòmic del Montsec para saber más sobre las actividades en el Montsec.



Figura 21: Vista panorámica del OAdM



Figura 22: Vista panorámica del OAdM



Figura 23: Fotografía del TJO

1.2.3. Calidad del cielo nocturno

La calidad astronómica del cielo del Montsec es bien conocida. Varias campañas de ensayos *in situ* llevadas a cabo a finales de los 90 confirmaron la calidad del lugar y su idoneidad para acoger un observatorio astronómico. Estas campañas se extendieron hacia los datos meteorológicos, la estabilidad del aire y la transparencia atmosférica. También se determinó que la contaminación lumínica en el OAdM es muy baja: el brillo medio del fondo del cielo en el zenith de las noches sin luna es de 22,0 magnitudes por segundo de arco cuadrado. Este valor indica que es un sitio particularmente oscuro.

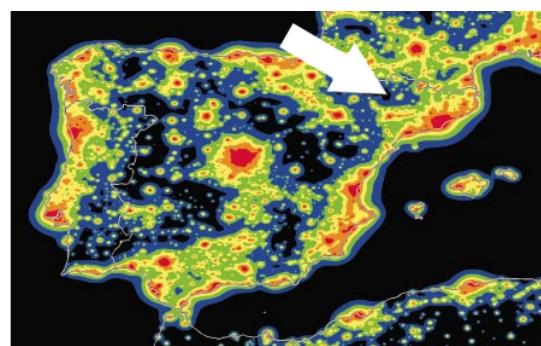


Figura 24: Brillo del cielo de la Península Ibérica (créditos en el punto 9 de la bibliografía)

Las variables climáticas y la estabilidad del aire están siendo monitorizadas continuamente desde el 2004 (el clima) y 2009 (estabilidad). Su evaluación confirma que las condiciones para la observación astronómica siguen siendo muy buenas.

1.3. Objetivos

El objetivo de este PFC es dar una solución a las necesidades de control robótico del TJO y del resto de elementos que intervienen en la operativa de funcionamiento de este telescopio. A continuación se expondrán detalladamente todo los elementos que lo componen, la manera en que interaccionan y cuales son las decisiones que se tienen que tomar.

Como se verá más adelante, el nuevo software de control propuesto hará énfasis en dar solución a las siguientes necesidades:

- Monitorización del entorno: debe ser capaz de tener en todo momento el estado del entorno, así como el estado de todos los elementos que forman este sistema (es decir, estado de la meteorología, estado del telescopio, estado de la cúpula, ...)
- Ejecución de tareas: debe encargarse de ejecutar las acciones para que la cúpula y el telescopio apunten a un objetivo, debe de hacer exposiciones (es decir, tomar fotos), debe de guardar los resultados obtenidos, ...
- Recuperación ante situaciones anómalas: debe ser capaz de detectar fallos que se produzcan mientras se está observando y debe en la medida de lo posible, intentar corregir esa situación.

En el inicio del proyecto, la operación del telescopio era presencial y remota, es decir, que se trataba de un telescopio robótico pero no desatendido. Para poder llegar al objetivo de robotizar el telescopio, se han tenido que modificar partes de software ya existentes y se han tenido que hacer mejoras, sobretodo en los interfaces de comunicación, para poder permitir tener control sobre el sistema y tener feedback del estado del mismo.

1.4. Estructura de la memoria

Esta memoria está estructurada en las siguientes secciones:

1. **Introducción:** Breve explicación del Observatori Astronòmic del Montsec (OAdM), del Telescopio Joan Oró (TJO) y de los objetivos de este PFC.
2. **Requerimientos científicos:** Modelo de funcionamiento del TJO.
3. **Sistema robótico:** Elementos que forman parte del sistema a controlar.
4. **Requerimientos del nuevo software de control:** Descripción de los requerimientos que debe cubrir el software.
5. **Estudio de mercado:** Busqueda de alternativas existentes para dar solución al problema.
6. **Diseño del nuevo software de control:** Explicación de la solución propuesta para conseguir los requerimientos planteados.
7. **Tareas previas a la implementación:** Explicación del trabajo previo realizado para poder aplicar el diseño propuesto.
8. **Implementación:** Explicación de como se ha programado la solución propuesta.
9. **Resultados:** Estado de la situación tras la migración al nuevo sistema de control.

10. **Planificación y costes:** Explicación del tiempo dedicado y del coste que ha tenido el proyecto.
11. **Conclusiones y trabajo futuro:** Estado final y posibles mejoras del sistema de control.
12. **Anexos:** Documentación adicional a este documento, como ficheros fuente, de configuración y explicación de las funciones desarrolladas.
13. **Bibliografía:** Lista de fuentes bibliográficas

Notas:

- Parte de los textos e imágenes presentados en las secciones 1, 2 y 3, han sido confeccionados por el personal que trabaja en los proyectos OAdM y TJO, desde el Institut d'Estudis Espacials de Catalunya (IEEC). Estos textos aparecen en la web www.oadm.cat, y en publicaciones, posters y presentaciones donde el equipo anterior ha participado.

2. Requerimientos científicos

El TJO tiene múltiples casos científicos por cubrir, incluyendo fenómenos transitorios. Este tipo de observatorio multipropósito requiere la capacidad para cambiar la programación del observatorio en tiempo real. La eficiencia se maximiza colocando la capacidad de toma de decisiones en un sistema de control diseñado para trabajar sin interacción humana.

2.1. Operaciones del telescopio

La mayoría de los procesos incluidos en las rutinas de observación científica y tareas de calibración se ejecutan sin interacción humana.

Las tareas admitidas por el observatorio y asumidas por el sistema de control robótico son:

- **Observaciones científicas.** Para realizar un caso científico específico, las propuestas son enviadas al observatorio en un conjunto de observaciones para obtener imágenes y datos.
- **Tareas de calibración.** Los procesos de calibración son necesarios para reducir los datos obtenidos en las observaciones científicas. Rutinariamente, se obtiene un conjunto nominal de imágenes de calibración por instrumento.
- **Seguridad.** A parte de realizar exposiciones, la seguridad del observatorio está supervisada a través de un conjunto de sistemas auxiliares responsables de monitorizar el entorno y las operaciones del hardware.
- **Mantenimiento.** Para mantener una arquitectura de software robusta y garantizar el flujo de datos, periódicamente se monitorizan y comprueban varios sistemas.

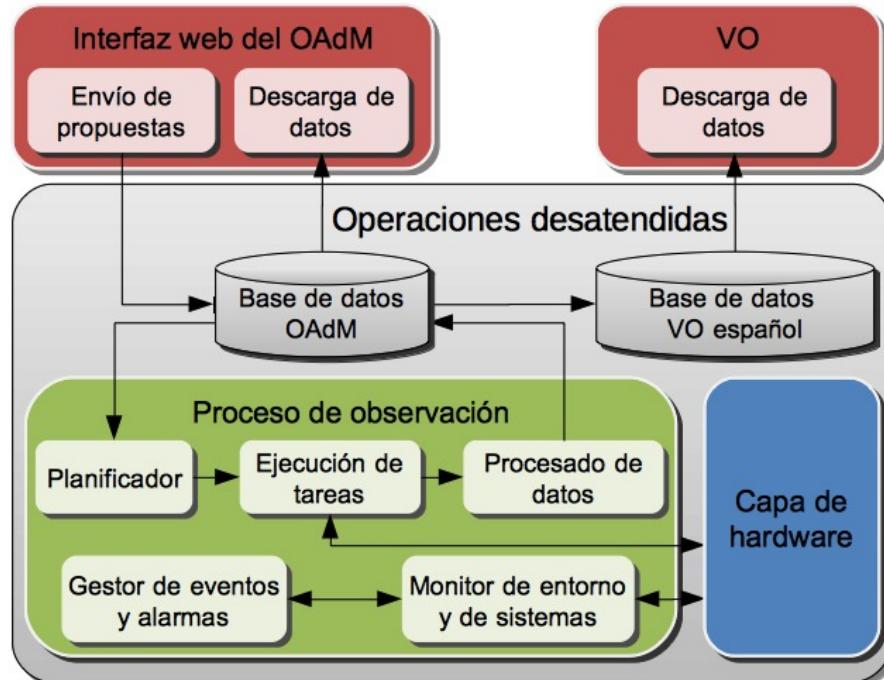


Figura 25: Operaciones del telescopio

2.2. Capas de control y flujo de trabajo

El sistema de control del observatorio trabaja de modo totalmente desatendido, o específicamente bajo control robótico, para la mayoría de los procesos en el flujo de datos. Sólo el envío de propuestas y la descarga de los datos requieren interacción humana. El sistema de control está basado en tres capas principales:

- **Capa de flujo de datos global.** Esta capa de software se encarga de ejecutar todas las rutinas de adquisición de datos, desde el envío de propuestas a la descarga de datos por parte del usuario, e incluye las respuestas del procesado de datos para la optimización de la planificación de las operaciones.
- **Capa de monitorización.** Esta capa de software se encarga de monitorizar la salud del sistema, de todos los sensores de entorno y dispositivos de soporte.
- **Capa física.** Contiene los elementos de hardware del sistema. Está compuesta por los dispositivos de ciencia para adquirir datos astronómicos (telescopio, cúpula e instrumentos), los sensores de entorno y otros dispositivos de soporte para fines de mantenimiento.

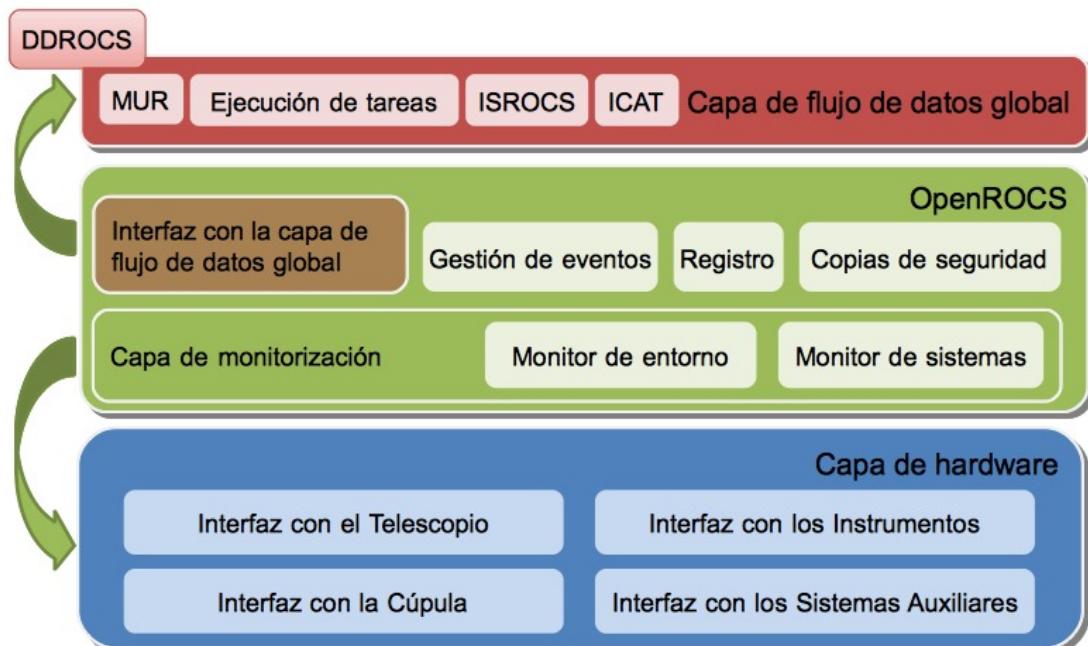


Figura 26: Capas de control y flujo de trabajo

2.3. Control de flujo de datos global

Se usan varias aplicaciones para lanzar los procesos necesarios para el control de las operaciones del observatorio: OpenROCS, ISROCS, ICAT i DDROCS. Todas juntas componen la suite de software llamada Robotic Observatory Control System (ROCS), diseñada y desarrollada de forma modular para adaptarse a diferentes configuraciones de hardware y, por tanto, a otros observatorios robóticos. Todas las aplicaciones se han desarrollado en el IEEC.

- **MUR.** Las observaciones científicas se introducen en el sistema mediante el envío de una propuesta al observatorio usando la aplicación web Management for Users in ROCS (MUR). El procedimiento para mandar una propuesta está descrito en la página de ayuda de MUR (en inglés).

- **OpenROCS.** El Open Robotic Observatory Control System (OpenROCS) es el elemento central del sistema de control. La ejecución de una observación científica o de una tarea de calibración incluye el seguimiento y control de todo el hardware involucrado (telescopio, instrumentación y cúpula), así como de los datos de entorno. Por lo que respecta al control del hardware, se incluye el apuntado y el seguimiento del objeto a observar, el posicionamiento del filtro y el enfoque, la adquisición de imágenes y las tareas de gestión interna que aseguran la integridad del sistema en caso de mal tiempo. OpenROCS se encarga de todos estos procesos con llamadas específicas a software (TALON, comandos SNMP, etc.). OpenROCS es un programador maestro de los procesos que el sistema tiene que ejecutar y un monitor del estado global del observatorio. Se encarga de controlar, a partir de unos eventos predefinidos, el flujo global de datos y los procesos de gestión interna. El objetivo es, por un lado, asegurarse que el sistema ejecuta las tareas rutinarias que maximicen el retorno científico y, por el otro, considerar todas las situaciones anómalas que podrían poner el observatorio en riesgo y activar las acciones correctivas o mitigadoras necesarias. Todas las aplicaciones en ejecución en el observatorio del TJO constituyen una capa independiente que se puede exportar a otros observatorios. OpenROCS se ha publicado bajo una licencia pública GNU/GPL.
- **DDROCS.** La base de datos del TJO, llamada Dynamic Database for the Robotic Observatory Control System (DDROCS), está dedicada a almacenar los detalles de las propuestas de una forma que optimice el flujo de datos des del envío hasta la recuperación de los datos por parte del usuario.
- **ISROCS.** Una rutina de planificación automática (ISROCS, proveniente de Intelligent Scheduler for the Robotic Observatory Control System) selecciona las tareas más adecuadas a realizar por el observatorio, de acuerdo a unos criterios predefinidos y con el objetivo de maximizar el tiempo dedicado a las operaciones científicas.
- **ICAT.** Cada vez que se adquiere una nueva imagen de ciencia o de calibración, se ejecuta un procedimiento en tiempo real para reducirla y analizarla. A continuación se realiza un control de calidad con el fin de determinar si la imagen se ha obtenido de acuerdo con los criterios predefinidos, y para proporcionar retorno al sistema de control del observatorio. Finalmente, las imágenes y los datos extraídos se guardan. El IEEC Calibration and Analysis Tool (ICAT) se desarrolló con todos estos objetivos y se usa para proporcionar datos de calidad a los usuarios.

3. Sistema robótico

El sistema robótico esta compuesto por 6 elementos básicos:

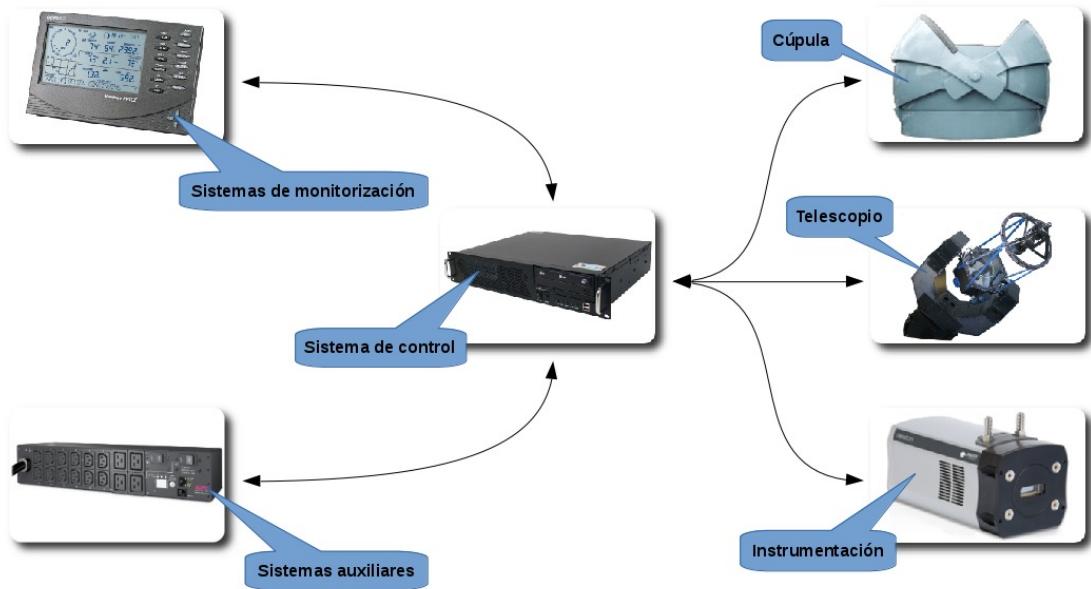


Figura 27: Elementos del sistema robótico

Como se puede ver en el diagrama, el sistema esta compuesto por:

- Telescopio.
- Cúpula.
- Instrumentación.
- Sistemas de monitorización.
- Sistema de control.
- Sistemas auxiliares.

3.1. Telescopio

El Telescopio Joan Oró (TJO), fabricado por OMI, tiene un espejo primario de 0.8 m de diámetro con un sistema óptico global de F/9.6 en configuración Ritchey-Chrétien. El telescopio tiene una montura ecuatorial de horquilla capaz de apuntar, en menos de 30 segundos (incluyendo la cúpula), a cualquier lugar por encima de 5 grados del horizonte.



Figura 28: Vista lateral del TJO

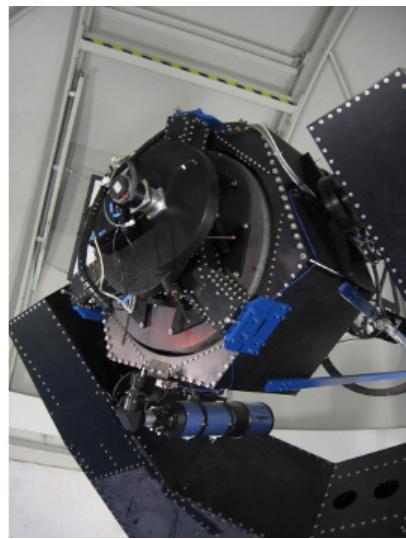


Figura 29: Vista trasera del TJO

El TJO tiene una precisión de apuntado mejor que un minuto de arco para la mayoría de posiciones, con los peores resultados a bajas alturas y en ángulos horarios grandes. El TJO no tiene guiado, pero el seguimiento es estable dentro de un segundo de arco para exposiciones de hasta cinco minutos, llegando a 10 minutos cerca del meridiano.

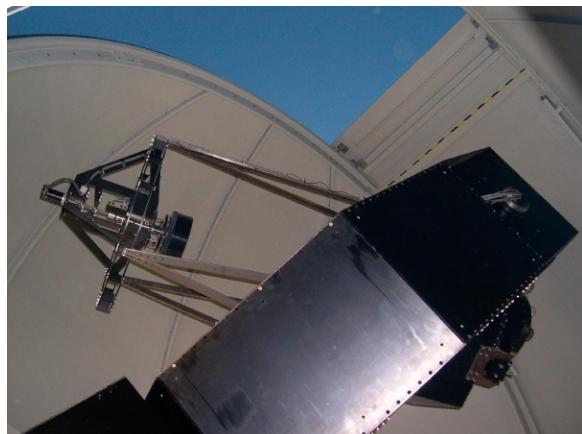


Figura 30: Vista lateral del TJO



Figura 31: Vista de la óptica del TJO

La instrumentación está ubicada en el foco Ritchey-Chrétien. Una cubierta segmentada protege el espejo primario del polvo cuando el telescopio está apagado. Todos estos elementos se controlan automáticamente.

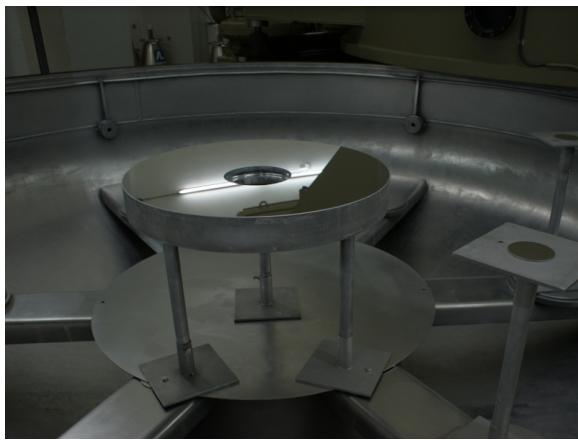


Figura 32: Fotografía del espejo primario del TJO

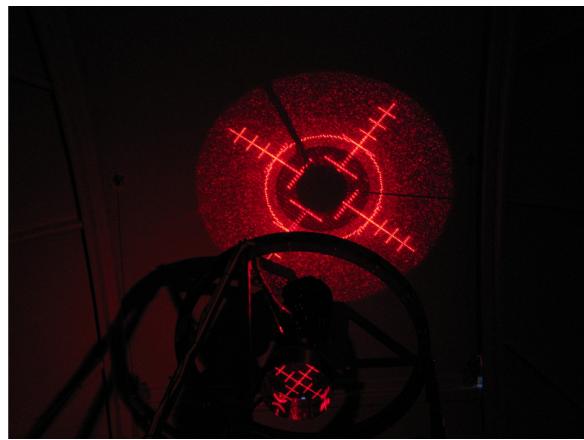


Figura 33: Fotografía de la colimación del TJO

La electrónica utilizada para controlar cada movimiento de los distintos elementos del telescopio (ejes de Ángulo horario y Declinación, cubierta del primario y enfoque), la rueda de filtros y la cúpula incluye las placas Clear Sky Institute Motion Controller (CSIMC), distribuidas por OMI.

3.2. Cúpula

El TJO está cubierto por una cúpula clásica de 6,15-m fabricada por Baader Planetarium GmbH. La visión del cielo del telescopio está habilitada a través de la apertura de una compuerta y un postigo, proporcionando una abertura de 97 x 29 grados cuadrados. La apertura y cierre de la cúpula y el movimiento en azimut están nominalmente controlados desde el ordenador principal utilizando el software TALON. La comunicación entre la tarjeta CSIMC y la electrónica de la cúpula se basa en una línea RS232.

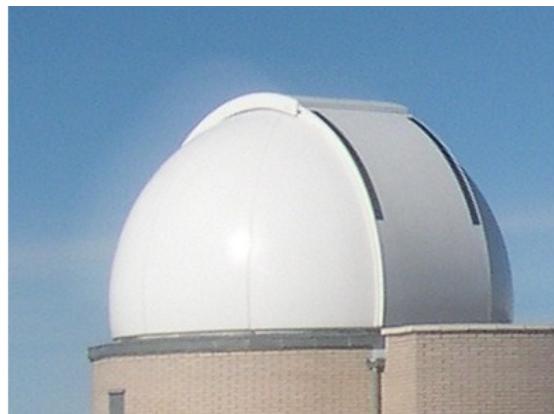


Figura 34: Cúpula del TJO

La solidez en el control de la apertura de la cúpula es una necesidad para garantizar el funcionamiento robótico y representa el punto más crítico de fallo en el TJO. La apertura de la cúpula puede ser controlada independientemente por un sistema de control redundante, proporcionando una respuesta de alta fiabilidad ante cualquier alarma imprevista y en caso de que el control nominal fallara.

3.3. Instrumentación

El TJO está equipado con dos instrumentos: una cámara de imágenes para fotometría (MEIA) y un espectrógrafo de resolución media (ARES). Sólo MEIA se encuentra actualmente disponible para la comunidad. ARES está empezando la fase de integración en el observatorio.

3.3.1. MEIA

MEIA (Medium-format Efficient Imager for Astronomy) es el instrumento de primera luz del TJO. Está compuesto por una cámara CCD de alta eficiencia y un juego de filtros Johnson-Cousins. Un sistema para la adquisición de flats de cúpula también está disponible.

El instrumento MEIA es la cámara óptica del TJO. Consiste en dos componentes: la cámara CCD y la rueda de filtros. Además, dispone de un sistema de flats de cúpula.

Cámara CCD

La cámara CCD es una ProLine 4240 (modelo PL4240-1-B), con un chip de $2k \times 2k$ iluminado por detrás, fabricada por Finger Lakes Instrumentation (FLI):

- Modelo: CCD42-40-1-B
- Fabricante del sensor: E2V
- Tipo de sensor: Back Illuminated
- Recubrimiento: Basic Midband
- Número de píxeles: 2048x2048
- Tamaño del píxel: $13.5 \times 13.5 \mu\text{m}$ (0.36×0.36 arcsec en el TJO)
- Campo de visión en el TJO: 12.3×12.3 arcmin
- Diagonal del sensor: 39.1mm
- Eficiencia cuántica pico: 96 %
- Temperatura de trabajo típica: -30°C
- Estabilidad en la temperatura: 0.1°C
- Ganancia típica: 1.53 e-/ADU
- Corriente de oscuridad (Dark) típico: $<1 \text{ e-}/\text{pixel/seg}$ a -30°C
- Error de lectura típico: 8 e- RMS a 500 kHz
- No linealidad: $<1 \%$
- Tiempo de lectura: 10 segundos



Figura 35: Cámara Finger Lakes Instrumentation

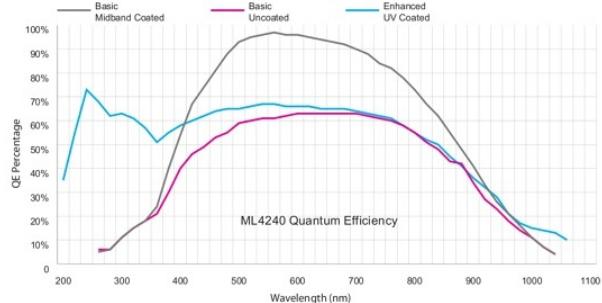


Figura 36: ML4240 Quantum Efficiency

Rueda de filtros

La rueda de filtros está físicamente acoplada al telescopio por la parte posterior del soporte del espejo primario. Tiene una capacidad de hasta 12 filtros de 3 pulgadas que se colocan en el eje óptico del telescopio girando la rueda de filtros. En la actualidad, hay cinco filtros fotométricos Johnson-Cousins (fabricados por Custom Scientific) instalados: U, B, V, Rc y Ic.

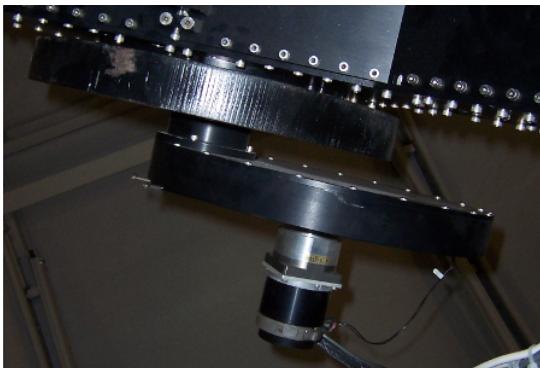


Figura 37: Rueda de filtros del TJO

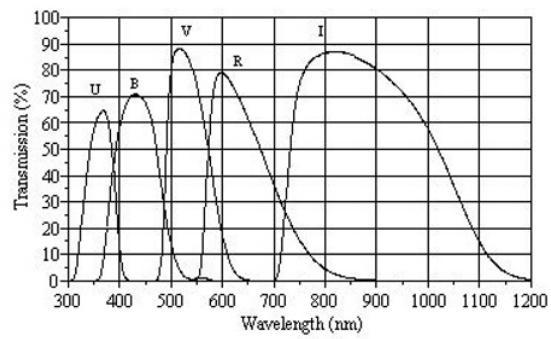


Figura 38: Respuesta de cada filtro

La magnitud límite de MEIA con un tiempo de exposición de 300 segundos (el máximo recomendado) para alcanzar una relación señal-ruido de 100 ($S/N=100$) para cada uno de los cinco filtros Johnson-Cousins es:

	U	B	V	Rc	Ic
	13.2	17.0	17.0	17.2	17.0

Tabla 1: Magnitud límite para una relación $S/N=100$

La relación señal-ruido (S/N) máxima que se puede conseguir con MEIA en una sola exposición depende del seeing y la luminosidad del cielo. Aun así, valores con $S/N=750$ deberían asegurar una S/N dentro del régimen de linealidad de la CCD, excepto en casos con muy buen seeing.

El tiempo necesario para cambiar de un filtro a otro es de 30 segundos.

Flats de cúpula

Aunque actualmente sólo se obtienen flats de cielo, se esperan obtener flats de cúpula de forma rutinaria con una homogeneidad mejor que el 1% (RMS) usando la siguiente configuración instrumental:

- Una pantalla blanca montada dentro de la cúpula.
- Un foco de luz de espectro completo que ilumina la pantalla.
- Un sistema de control para encender y apagar la luz cuando lo necesite el sistema de control del observatorio.



Figura 39: Lampara para hacer flats de cúpula



Figura 40: Pantalla para hacer flats de cúpula

Secuencia de calibración estándar

El sistema de control del TJO está diseñado para obtener imágenes de calibración de forma periódica, incluyendo bias, darks y flats de cielo. Estas imágenes sirven para corregir los errores de la electrónica de la cámara y neutralizar el ruido generado por la temperatura:

- Bias: Las imágenes de bias se obtienen realizando exposiciones de 0 segundos y sirven para obtener el patrón de electrones que tiene la cámara en el momento de iniciar una captura. Se puede decir que estas imágenes indican los valores de pixels que cada imagen tendrá como mínimo. Normalmente se obtienen en múltiples de 5 después de la adquisición de los flats de cielo o entre dos secuencias de ciencia a lo largo de la noche. Normalmente, se proporciona a los observadores un mínimo de diez bias.
- Dark: Las imágenes de dark o de corriente de oscuridad se obtienen realizando exposiciones de n segundos sin abrir el shutter. De esta manera, se obtiene el patrón de electrones de la cámara para una imagen que supuestamente, no debería de tener cuentas. Normalmente se obtienen justo antes o después de las imágenes de bias. El tiempo de exposición de las imágenes de dark depende del tiempo de exposición de las imágenes de ciencia. Normalmente, el tiempo de exposición de las imágenes de dark corresponde al tiempo máximo de exposición de las imágenes de ciencia.
- Flats de cielo: Las imágenes de aplanamiento de campo (flats de cielo) son imágenes que se obtienen realizando exposiciones de n segundos con el shutter abierto y apuntando a una zona del cielo o pantalla de luminosidad uniforme. Sirven, junto con los darks para obtener el rango de cada pixel y poder hacer así, la corrección de las exposiciones reales a targets que se deseen observar. Se obtienen rutinariamente

apuntando a varios campos vacíos durante el crepúsculo. Cuando las condiciones de cielo no permiten adquirir flats de cielo, se proporcionan imágenes obtenidas anteriormente a los observadores. Las imágenes de aplanamiento de campo se obtienen rutinariamente con una iluminación que proporciona una homogeneidad mejor que el 1% de dispersión (RMS).

Se espera que las exposiciones de calibración sean usadas para una evaluación al vuelo de la calidad de las imágenes de ciencia con MEIA. Todas las imágenes de calibración se proporcionan a los observadores que tienen imágenes de ciencia durante la noche.

3.3.2. ARES

ARES (Astronomical mid-REsolution Spectrograph) es un espectrógrafo alimentado por fibra, situado en una sala dedicada en la planta baja del observatorio. Los principales programas científicos que se pueden llevar a cabo con ARES incluyen la observación de estrellas relativamente brillantes a resolución intermedia.



Figura 41: Óptima y VPHs del instrumento ARES

ARES es el espectrógrafo óptico (diseñado por Fractal SLNE) que se está instalando actualmente en el TJO. Principalmente, está concebido para obtener espectroscopía de resolución media con las características siguientes:

- Magnitud límite en estrellas: $V < 11$ mag.
- Transmisividad global: $> 10\%$
- Resolución espectral: $R=12000$
- Dos ventanas espectrales:
 - Verde: entre 495 y 529 nm
 - Rojo: entre 634 y 678 nm

Para alcanzar la alta transmisividad global ($>10\%$), dos telescopios (fabricados por Takahashi) y unas VPH se han colocado en configuración de Littrow.

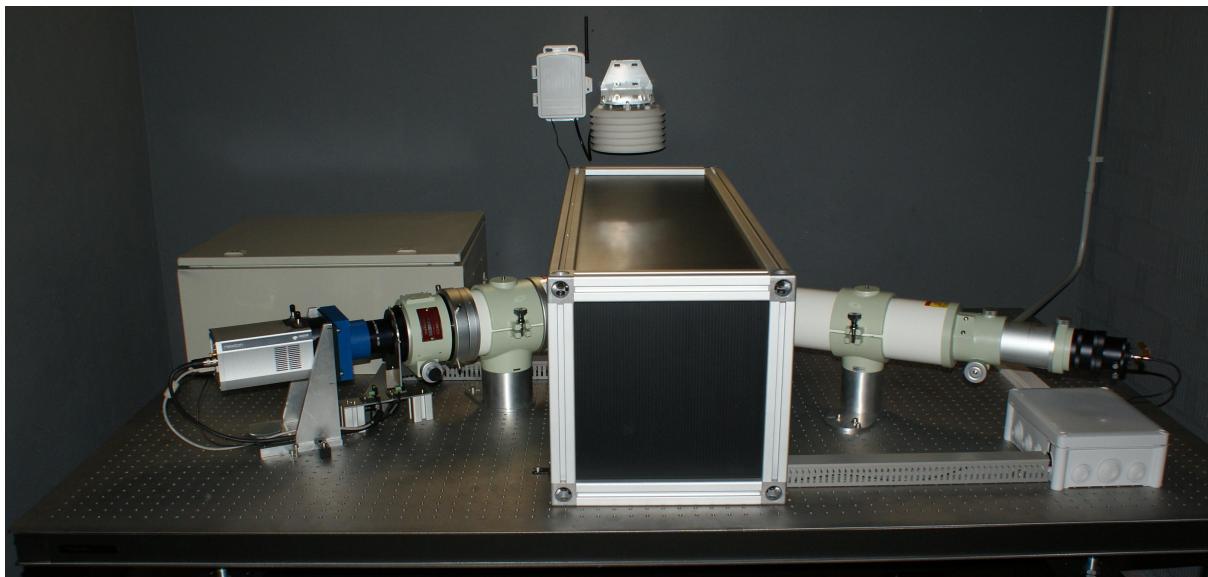


Figura 42: Sistema completo del instrumento ARES

VPHs

El sistema de dispersión de ARES está compuesto por dos VPH fabricadas por Wasatch Photonics, proporcionando las dos ventanas espectrales. Las dos VPH están montadas sobre un carro (preparado para sostener hasta tres VPH), permitiendo un cambio rápido de una VPH a otra.

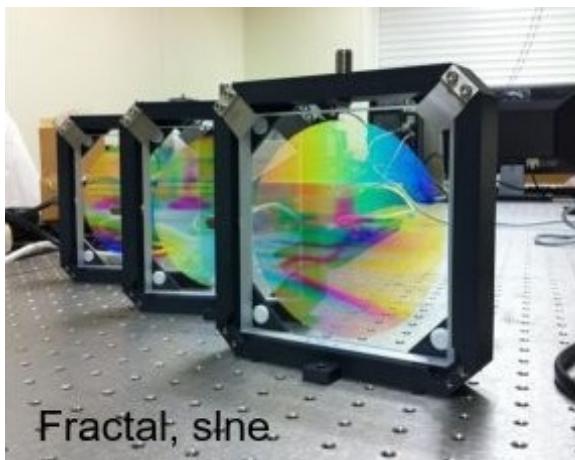


Figura 43: VPHs de ARES sin guía

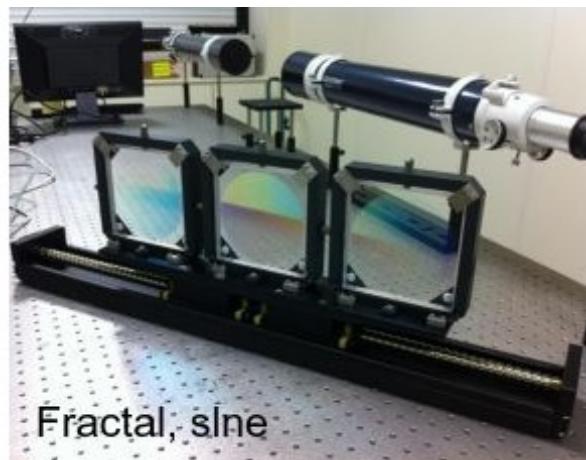


Figura 44: VPHs sobre la guía

Detector CCD

El detector usado es una cámara CCD Newton DU940P de Andor Technology con las siguientes características:

- Tipo de sensor: BV- Back Illuminated CCD, VIS optimized
- Píxeles activos: 2048 x 512

- Tamaño del píxel: $13.5 \times 13.5 \mu\text{m}$
- Área de la imagen: $26.7 \times 6.9 \text{ mm}$
- Máximo enfriamiento: -100°C
- Ruido de lectura: hasta un mínimo de 2.8 electrones
- Corriente de oscuridad (Dark): hasta un mínimo de 0.0002 e-/píxel/seg
- Linealidad: superior al 99 %



Figura 45: Cámara Andor DU940P

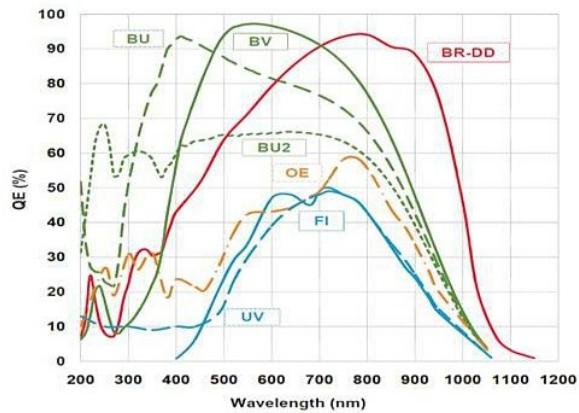


Figura 46: DU940P Quantum Efficiency

3.4. Sistemas de monitorización

Una monitorización fiable es uno de los aspectos más críticos para la seguridad de las operaciones robóticas. Ésta implica un conocimiento en tiempo real del entorno y una alta fiabilidad en el hardware, aspectos que se han tratado en el TJO con el equipamiento adecuado.

3.4.1. Dispositivos para la monitorización meteorológica

Varios sensores proporcionan una monitorización fiable de las variables meteorológicas que son críticas para la seguridad de las operaciones del TJO. Se aplica redundancia en la mayoría de los sistemas (humedad, velocidad del viento, sensor de lluvia, etc.), lo que garantiza decisiones seguras para preservar la integridad del sistema.

- Previstorm. Este sensor, fabricado por INGESCO, registra las variaciones del campo electrostático para predecir tormentas.



Figura 47: Sensor externo previstorm



Figura 48: Electrónica control previstorm

- EM-Davis. Es una estación meteorológica Davis Vantage Pro II con cuatro nodos distintos. Un nodo está ubicado fuera del edificio del TJO y está equipado con sensores para medir el viento, la humedad, la temperatura y la presión atmosférica. Los otros tres nodos proporcionan datos de temperatura y humedad y están ubicados: en la cúpula del TJO, en la sala de ARES y en la sala de control.



Figura 49: Estación meteorológica DAVIS Vantage Pro II

- EM-SMC. Es una estación meteorológica Campbell Scientific que pertenece al Servei Meteorològic de Catalunya (SMC), propiedad de la Generalitat de Cataluña, y ubicada en el recinto del OAdM.



Figura 50: Estación meteorológica Campbell Scientific del SMC

- EM-TFRM. Es una estación meteorológica Vaisala que pertenece al Telescopi Fabra-ROA Montsec (TFRM) y que está ubicada en el recinto del OAdM.



Figura 51: Estación meteorológica Vaisala del TFRM

- Sensor de lluvia. El sensor de lluvia está ubicado cerca de la cúpula del TJO. Es un modelo IRSS88 fabricado por Eigenbrodt.



Figura 52: Sensor de lluvia IRSS88

- Sensor de nubes Boltwood. El Boltwood Cloud Sensor II está ubicado cerca de la cúpula del TJO y proporciona información diferencial de la temperatura de brillo del cielo (y, por tanto, de la nubosidad).



Figura 53: Sensor de nubes Boltwood Cloud Sensor II

3.4.2. Dispositivos para la monitorización de tiempo

El sistema usa antena GPS (Garmin) para la determinación de la base de tiempo.



Figura 54: Antena GPS (Garmin)

Estos datos son correlados con otros servidores de tiempo usando el protocolo NTP.

3.5. Sistema de control

Como primera aproximación, se desarrolló un software llamado OpenROCS v1.0, que realizaba las siguientes tareas:

- Monitorización del entorno: se encarga de leer los datos de la EM-Davis, EM-SMC y del sensor de lluvia (raindetector). Estos datos son usados por la aplicación para publicarlos en un servicio tipo telnet en el puerto 6666 y para generar los ficheros de entrada que usa Talon.
- Iniciar el software Talon para poder empezar a operar con el telescopio (usando xobs y xcamera como interfaz gráfica de Talon). En este punto, hay que remarcar que la única tarea que realizaba era ejecutar un script de TALON llamado boot, pero sin ningún tipo de feedback sobre lo que sucedía posteriormente.

3.5.1. OpenROCS v1.0

Inicialmente, se desarrolló OpenROCS v1.0 como software para automatizar el control del telescopio, pero rápidamente, surgieron nuevos requerimientos que plantearon volver a hacer este software teniendo en cuenta las nuevas necesidades.

OpenROCS v1.0, es un software diseñado como una aplicación distribuida. Para ello, se empleó la librería ICE, que permite comunicar diferentes procesos entre diferentes máquinas y usando lenguajes de programación diferentes.

ICE es una alternativa moderna para crear instancias a objetos tales como CORBA o COM/DCOM/COM+. Es fácil de aprender, sin embargo, proporciona una infraestructura de red de gran alcance para las exigentes aplicaciones técnicas. Cuenta con un lenguaje de especificación orientada a objetos, fácil de usar C++, C#, Java, Python, Ruby, PHP, y Visual Basic, un protocolo muy eficiente, métodos de invocación y dispensar asíncronos, TCP/IP, seguridad UDP/IP basada en SSL, una firewall de seguridad, y mucho más.

3.5.2. Software de control TALON

El equipo básico suministrado por el fabricante del telescopio incluía el software TALON, escrito en C y shell scripts que se ejecutan en GNU/Linux. TALON, que ahora se distribuye gratuitamente bajo la licencia

GPL-2.0 y que fue desarrollado por Elwood Downey, permite el control de distintos elementos para adquirir automáticamente observaciones astronómicas. Los elementos de hardware controlados por TALON en el TJO son los siguientes: telescopio, cúpula y MEIA.

TALON es una solución completamente automatizada de código abierto para el control de telescopios automatizados y del observatorio. TALON controla todos los aspectos de las observaciones robóticas astronómicas, incluyendo el control del telescopio, control de cúpula, procesamiento de imágenes y operaciones programadas.

De TALON, se pueden distinguir 2 tipos de procesos:

- Demonios: son procesos que se ejecutan en background y son los que acceden de forma directa con el hardware mediante RS232 y USB, dependiendo del dispositivo que controle.
- Interfaces gráficos: son programas que permiten al usuario interactuar con los demonios y controlar así el telescopio.

Todos los procesos anteriores se comunican entre si mediante pipes con nombre y zonas de memoria compartida (shared memory). Este software esta pensado para ser empleado desde los interfaces gráficos y por lo tanto, tiene que ser un operador de telescopio el que debe ejecutarlo para realizar observaciones. Como nota, indicar que es posible controlar 100 % el telescopio enviando comandos a las pipes con nombre y esperando respuesta de las mismas.

Interfaz xobs

xobs es la ventana principal de control de talon y se ejecuta con el comando startTel. Contiene todas las herramientas necesarias para la operación de monitoreo y calibración. Esta ventana proporciona un control manual del telescopio y todos los periféricos conectados, tales como una rueda de filtro o control de cúpula. También proporciona una visualización constante de la posición actual del telescopio, según los cálculos de la retroalimentación de los encoders de motor. Esta información se proporciona en un conjunto de cuadros de texto dentro de la ventana xobs.

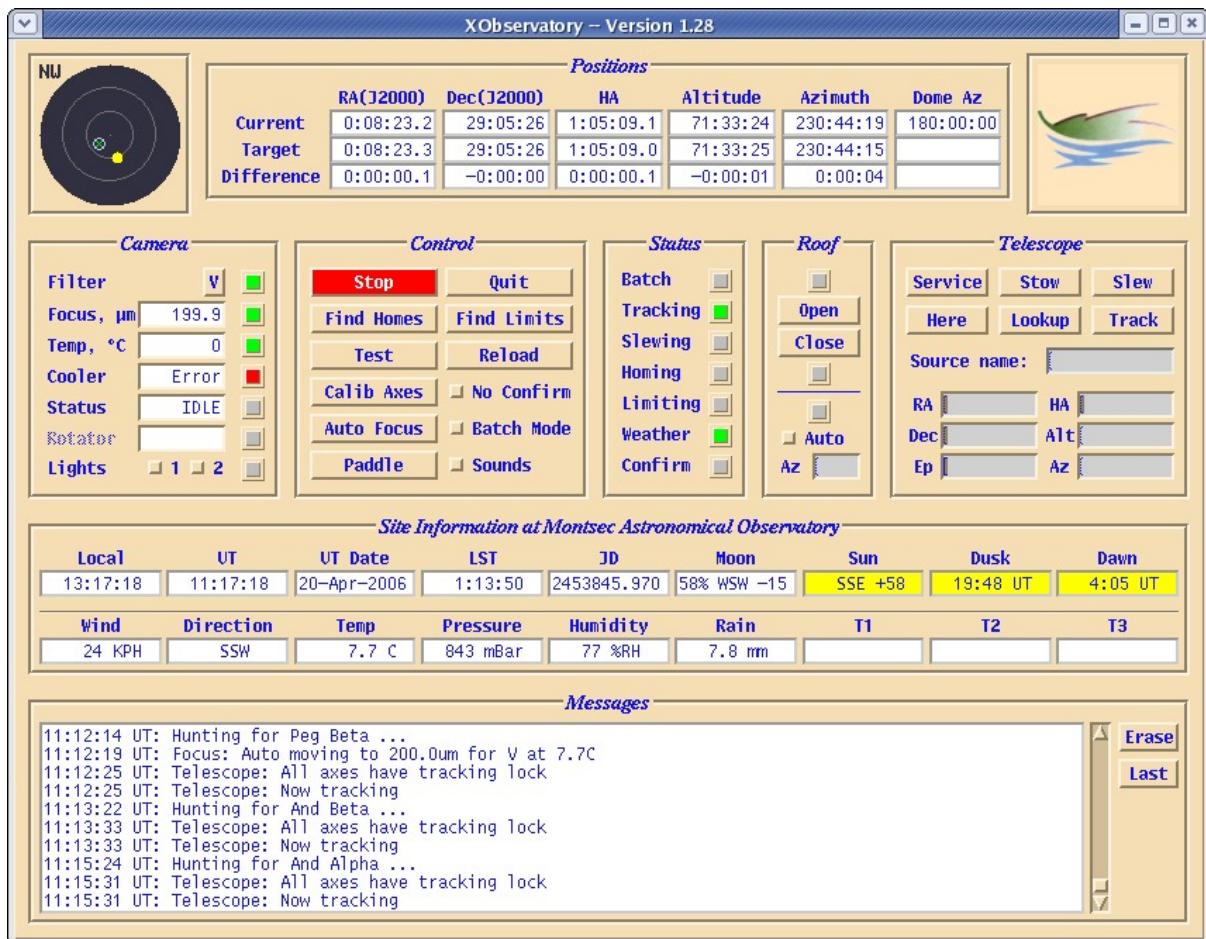


Figura 55: Interfaz xobs de TALON (Main Window)

Usando el comando paddle en la ventana xobs, el usuario puede colocar el telescopio, rueda de filtros y la posición de enfoque manualmente. El movimiento del telescopio hacia el este y el oeste se conoce como la ascensión recta (AR) o ángulo horario (HA) del telescopio. Para mover de norte a sur se conoce como Declinación (Dec).

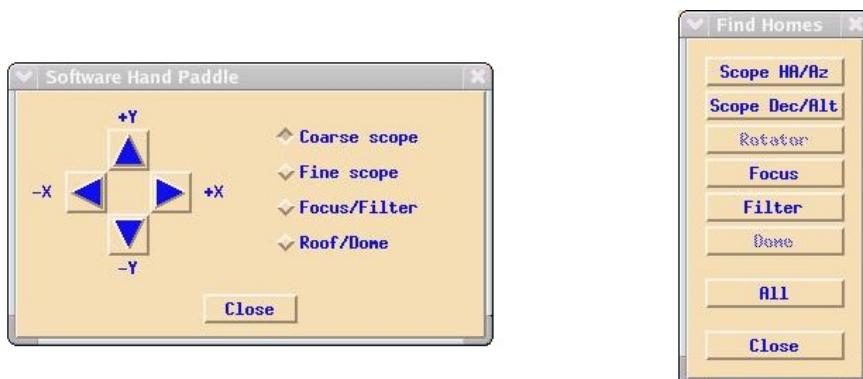


Figura 56: Interfaz xobs de TALON (Software Hand Paddle y Find Homes)

Interfaz xcamera

xcamera es otra aplicación de la suite de Talon. Proporciona un control total sobre las funciones de una cámara CCD conectada al telescopio.

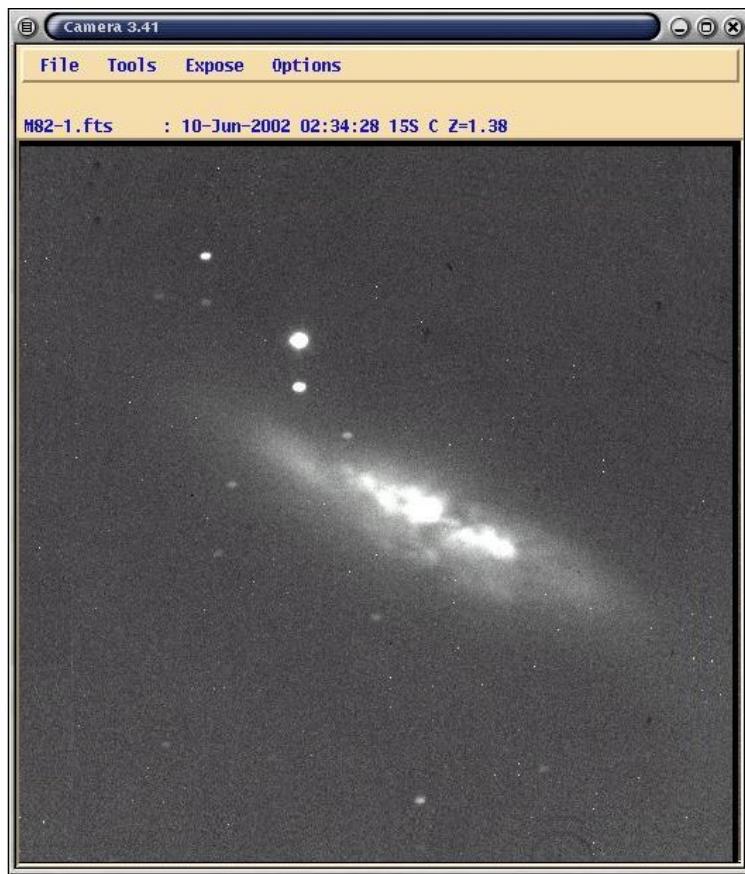


Figura 57: Interfaz xcamera de TALON

La aplicación de la cámara incluye herramientas para controlar el tiempo de exposición, tamaño de la imagen, software de filtrado de imágenes y análisis de imágenes. xcamera también contiene herramientas para ajustar el brillo y el contraste de las imágenes, la determinación del área de interés (AOI) de la imagen y también puede etiquetar automáticamente los objetos mediante la comparación con el Sistema Mundial de Coordenadas (WCS). Esta última herramienta es, de hecho, un algoritmo de coincidencia de patrón que permite al sistema comparar los patrones conocidos de objetos de la base de datos de WCS.

Interfaz telsched

telsched es el elemento de Talon que hace posible las sesiones robóticas de observación desatendidas. Esto puede ser una función crítica para las instituciones que realizan investigaciones desde ubicaciones remotas o las que requieren observaciones repetidas de objetos particulares durante un período determinado de tiempo.

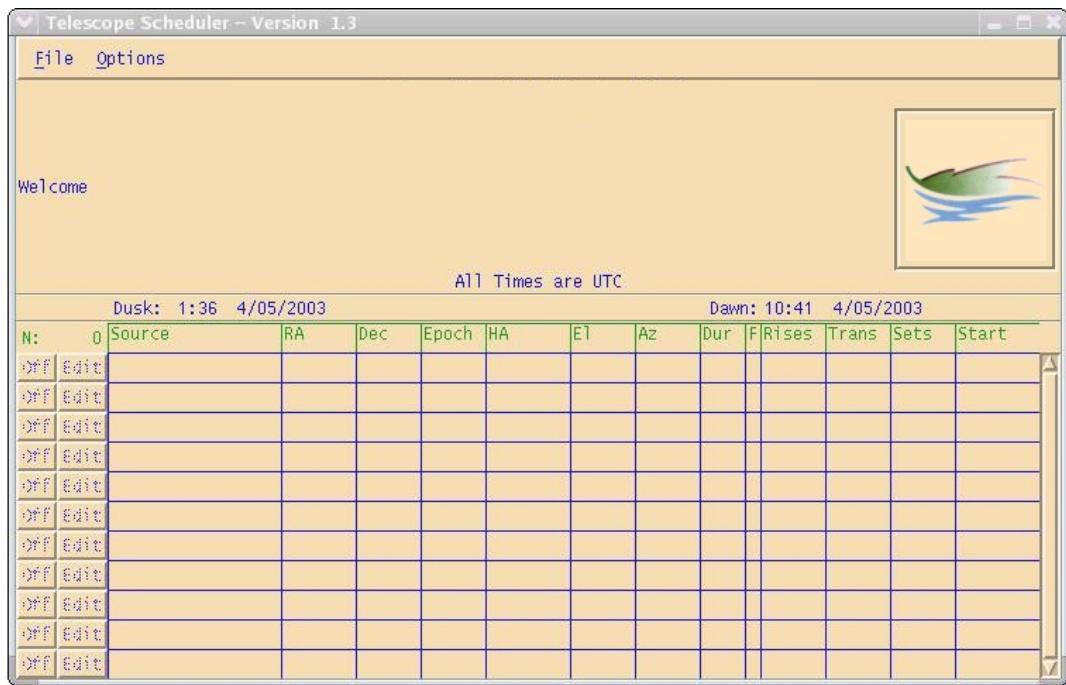


Figura 58: Interfaz telsched de TALON

Interfaz XEphem

XEphem proporciona un software de efemérides o interfaz gráfico para mostrar el cielo al resto de la suite de Talon. Al igual que con otras efemérides, depende en gran medida de tener las correctas coordenadas geográficas y de tiempo. La información de posición puede ser configurada manualmente por el usuario. XEphem también se pueden configurar para usar un GPS a intervalos regulares para el ajuste de la hora del sistema.

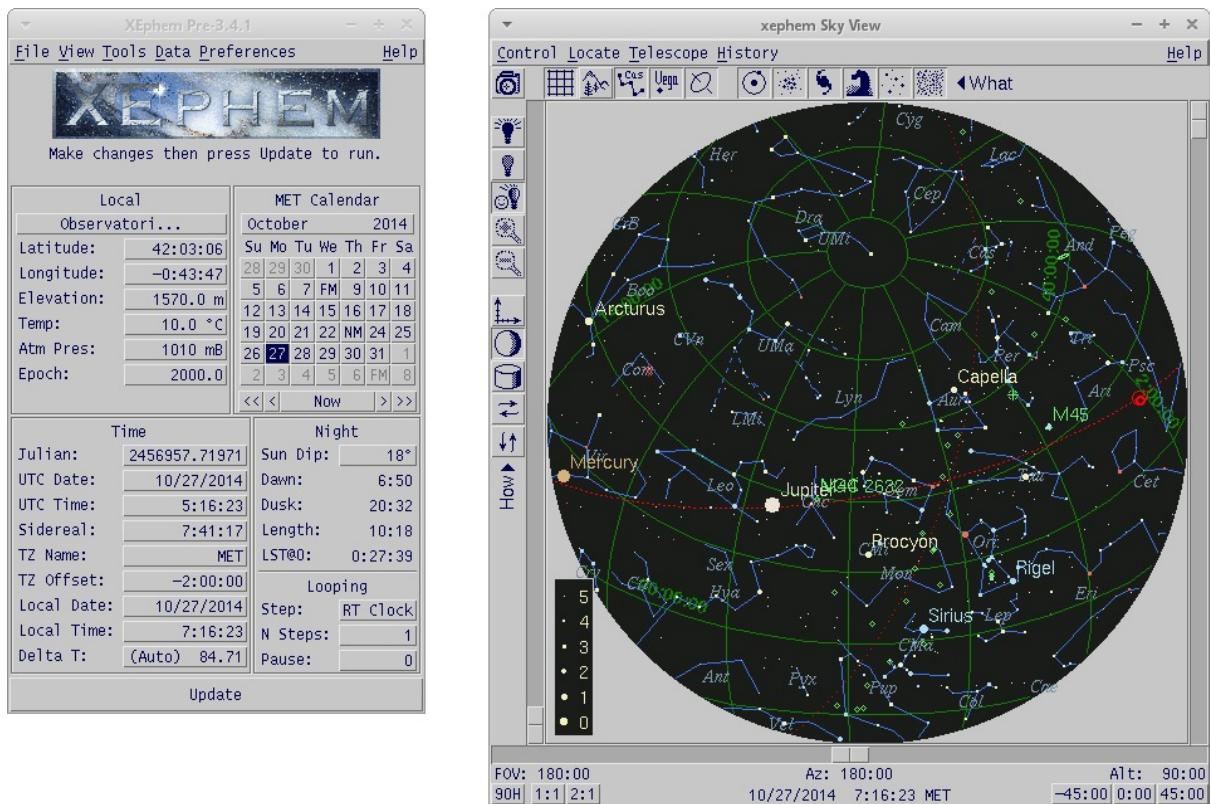


Figura 59: Interfaz XEphem

El programa XEphem, lanzado desde la línea de comandos con `xephem`, proporciona una visión granular del cielo actual. Los datos sobre cada objeto es ofrecida en una pantalla emergente con el botón derecho. El usuario también puede apuntar el telescopio usando este pop-up, una característica ampliamente utilizada para la calibración. Como una alternativa a zoom, el usuario puede seleccionar un umbral de magnitud mínima (brillo aparente). Esto permite estrellas más brillantes para ser filtrados en la vista de efemérides, dejando sólo los objetos más tenues en la ventana. La vista del cielo también se puede girar, y se proporciona filtrado de tipo de objeto. Por ejemplo, los cúmulos globulares pueden ser seleccionados, eliminando la vista de todos los otros tipos de objetos.

3.6. Sistemas auxiliares

3.6.1. Alimentación eléctrica

El OAdM se alimenta directamente de la red eléctrica de la compañía Endesa. La instalación también dispone de un generador de emergencia de 35-kW que proporciona suficiente energía para que el sistema pueda continuar las operaciones, incluso en caso de un fallo del corriente eléctrico.



Figura 60: Placas solares



Figura 61: Baterías

Varios Sistemas de Alimentación Ininterrumpida (SAIs), usando tecnología de doble conversión, también están instalados en el TJO para proporcionar alimentación eléctrica al los equipos que son muy sensibles a sobre-tensiones y que requieren un aislamiento eléctrico: instrumentación astronómica, sensores, ordenadores y cúpula.



Figura 62: Control de carga de baterías



Figura 63: Cuadro eléctrico general

La utilización de varias Unidades de Distribución de la Corriente (UDCs) permiten el control automático o remoto de la alimentación eléctrica de varios dispositivos. Los SAIs y los UDCs distribuyen la energía a todos los dispositivos y aseguran la alimentación durante el tiempo suficiente, incluso en el caso de un fallo del suministro eléctrico.



Figura 64: Depósito de combustible



Figura 65: Generador eléctrico diesel

3.6.2. Aislamiento eléctrico

La ubicación del OAdM lo hace propenso a tormentas eléctricas durante los meses de verano. La protección contra las sobre-tensiones es un requerimiento para asegurar la fiabilidad de un observatorio robótico como el TJO.

La instalación del TJO está adecuadamente protegida contra impactos de relámpagos y sobre-tensiones inducidas mediante diferentes tipos de filtros (Ethernet, RS232, etc.). Se usan filtros para los puertos serie estándar y para las conexiones de red, y elementos opto-acoplados (o sea, transmisión a través de fibra óptica) para el resto. También se usan tres tipos de filtros para la distribución de las líneas eléctricas, dependiendo del daño potencial que una sobre-tensión grande podría provocar en el dispositivo.

3.6.3. Internet y LAN

La comunicación a través de Internet es otro elemento a tener en cuenta para la seguridad del observatorio. Se usa una conexión de fibra óptica de 100 Mbps de ancho de banda para las comunicaciones externas. Ésta conecta el observatorio con la Anella Científica, una red de comunicaciones de alta velocidad (de 2 Gbps de ancho de banda) gestionada por el CSUC (Consorci de Serveis Universitaris de Catalunya), y que conecta universidades y centros de investigación de Cataluña. La red está conectada a RedIRIS, la red de investigación española. A través de RedIRIS, se llega a las redes internacionales más avanzadas. La red se monitoriza de forma continua desde el CSUC y también por el sistema de control del observatorio.



Figura 66: Zona superior armario de comunicaciones



Figura 67: Zona inferior armario de comunicaciones

4. Requerimientos del nuevo software de control

Para poder dar solución al problema, es necesario un sistema que realice las siguientes tareas:

- Monitorizar las variables de entorno (meteorología, gps, ...)
- Tomar decisiones (encender fuentes de alimentación, abrir o cerrar la cúpula, realizar exposiciones con diferentes configuraciones instrumentales, ...)
- Comunicarse con los diferentes dispositivos (telescopio, cúpula, cámaras, PDUs, SAIs, ...)
- Intercambiar información con otros equipos (ejecutar comandos remotamente, compartir información, ...)

Pero además, debe cumplir las siguientes restricciones:

- Debe permitir evolutivos de forma fácil, rápida y fiable. El entorno donde se ejecutará esta orientado a la investigación y liderado por científicos cuyas necesidades cambian a diario dependiendo de los avances que realizan en sus investigaciones. Todo esto implica que debe permitir que un usuario con un cierto nivel en áreas de ciencia y informática (como un operador de telescopio) puedan configurar el comportamiento del sistema completo.
- Dado que en caso de avería, quizás no existan piezas de recambio y deban de ser reemplazados por sistemas nuevos con nuevos interfaces, en la medida de lo posible, el sistema debe ser lo suficientemente flexible como para adaptarse a este tipo de situaciones.
- Debe ser lo suficientemente automático como para que se ejecute de forma desatendida. Esto implica que debe ser fiable como para poder prescindir de un operador de telescopio.
- Dado que existe software que realiza tareas complejas (control del telescopio y cúpula, realización de exposiciones, control de tiempo mediante GPS, control de entorno mediante SNMP), no es necesario que se integren estas funcionalidades en el nuevo sistema de control, sino que mediante interfaces de comunicación pueda ser ejecutadas y así, reciclar parte de la programación ya existente.

5. Estudio de mercado

Antes de empezar a pensar en el diseño de una solución, lo ideal es poder realizar un estudio de mercado para ver que software existe y como nos puede ayudar en el diseño de la solución.

En este apartado, se ha hecho la labor de buscar información de algunas de las soluciones que existen, así como identificar sus prestaciones y características.

5.1. RTS2

RTS2, o Remote Telescope System, 2nd Version, es un paquete de código abierto para el control remoto integrado de un observatorio bajo el sistema operativo GNU/Linux. Sus objetivo es la de controlar un observatorio completo desde un ordenador, poder realizar apuntado y seguimiento y tomar imágenes.

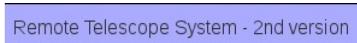


Figura 68: Logo del software de control de observatorios RTS2

RTS2 está diseñado para gestionar un observatorio en modo totalmente autónomo, recogiendo los targets de una base de datos, almacenando los datos en la base de datos, procesando imágenes y guardando las coordenadas WCS en la base de datos y ofreciendo acceso al Observatorio Virtual. En la actualidad se está ejecutando en diversos telescopio en todo el mundo. Para el control de dispositivos de varios fabricantes, han desarrollado una capa abstracta que permite el control de todas las posibles combinaciones de los controladores de monturas, CCDs, fotómetros y cúpulas.

Las características actuales incluyen:

- Integración de astrometría online utilizando el paquete astrometry.net o cualquier otro paquete de astrometría de línea de comandos.
- Creación de targets desde SIMBAD.
- Targets de observación para la luna, planetas, planetas enanos y objetos del sistema solar.
- Módulo para la recepción y el procesamiento de las alertas de GCN.
- Observación GRB controlada por scripts, que puede depender del tiempo desde el evento de GRB.
- Base de datos PostgreSQL, que almacena todos los detalles de la observación, que incluye imágenes con coordenadas WCS.
- Monitoreo basado en ncurses (RTS2-mon), que le permite controlar el progreso de observación.
- Servidor INDI habilitado para la interacción con el telescopio.
- Función de prioridad (selector), planificado (night schedule), manual (rts2-mon) y observaciones urgentes (GRBs).
- Arquitectura plug-and-play para sensores meteorológicos (lluvia, nubes, viento, ..). Cada sensor puede reportar mal tiempo, y usted puede especificar qué sensores son necesarios para operar el observatorio.

Tienen la intención de añadir a RTS2 siguientes características:

- Integración con el Horizon Service (JPL-NASA) para los parámetros de los pequeños cuerpos del sistema solar.

- Targets como satélites que orbitan la Tierra.
- Interfaz para la base de datos de imágenes VO.
- Red cliente-servidor Jabber para el intercambio de solicitudes de observación.
- Enviar mensajes de texto a un cliente Jabber, para conocer lo que está haciendo el telescopio.
- Interfaz gráfica de usuario para la gestión de los targets de observación.
- Programación de algoritmos genéticos (basado en este trabajo: <http://lascaux.asu.cas.cz/~petr/proyecto/scheduling.pdf>).

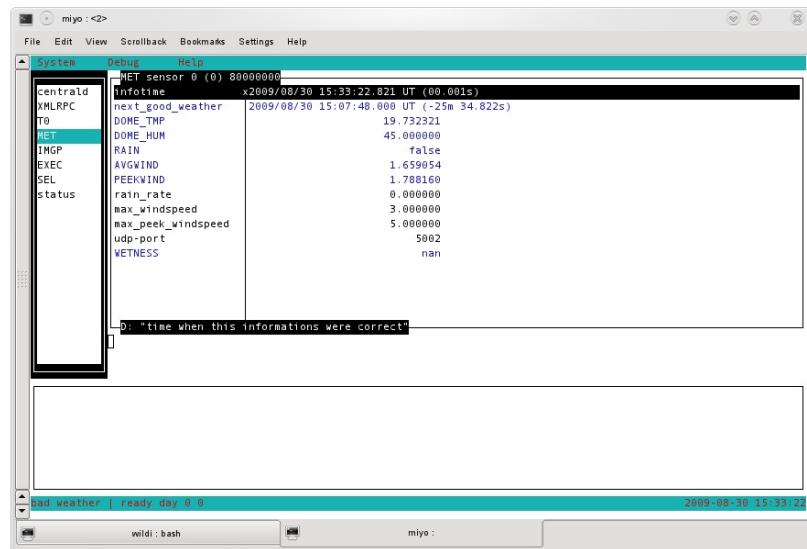


Figura 69: Interfaz de usuario de RTS2

Este software esta liberado bajo la licencia GPL-2.0.

5.2. ASCOM

Un driver ASCOM actúa como una capa de abstracción entre el cliente y el hardware eliminando así cualquier dependencia del hardware en el cliente, y haciendo que el cliente sea automáticamente compatible con todos los dispositivos que soportan las propiedades y métodos mínimos requeridos. Por ejemplo, esta abstracción permite a un cliente ASCOM utilizar un dispositivo de imágenes sin necesidad de saber si el dispositivo está conectado a través de una conexión en serie o de red.

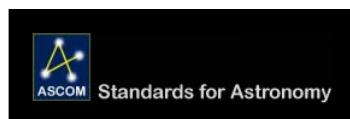


Figura 70: Logo del paquete de software ASCOM

ASCOM define un conjunto de propiedades y métodos necesarios que el software compatible ASCOM puede utilizar para comunicarse con un dispositivo compatible con ASCOM. También define una serie de propiedades y métodos opcionales para aprovechar las características comunes que pueden no estar disponibles para el dispositivo de cada fabricante. Al poner a prueba diversas propiedades de una aplicación cliente ASCOM puede determinar qué funciones están disponibles para su uso.

Propiedades y métodos son accesibles a través de interfaces de secuencias de comandos, lo que permite el control de dispositivos de aplicaciones de secuencias de comandos estándar como VBScript y Javascript. De hecho cualquier idioma que admita el acceso a los objetos COM de Microsoft puede interactuar con ASCOM.

El paquete de software de la plataforma ASCOM está disponible para su descarga y instala las bibliotecas comunes y documentación, así como una colección de drivers ASCOM para una amplia gama de equipos. Los drivers adicionales ASCOM para dispositivos no incluidos en el paquete ASCOM puede ser descargados e instalados por separado.

Aunque ASCOM se utiliza predominantemente por la comunidad de aficionados, porque la norma está disponible libremente, también se utiliza en algunas instalaciones profesionales.

Este software esta liberado bajo la licencia Creative Commons.

5.3. TALON

Este es el software que actualmente se usa para controlar el TJO (véase el capítulo 2.1)



Figura 71: Logo del software de control de observatorios TALON

Este software esta liberado bajo la licencia GPL-2.0.

5.4. INDI

Indi, o Instrument Neutral Distributed Interface es un protocolo de Sistema de Control Distribuido para permitir el control, adquisición de datos y el intercambio entre los dispositivos de hardware y front ends, con especial énfasis en la instrumentación astronómica:



Figura 72: Logo del sistema de control distribuido INDI

- Open source: INDI se distribuye bajo la Licencia Pública General Reducida de GNU (LGPL v2 +), puede utilizar y modificar el código fuente para adaptarse a sus necesidades a cualquier entorno.
- Cross Plataforma: INDI es una biblioteca que implementa el protocolo INDI para sistemas operativos POSIX. Actualmente, Linux, BSD y OSX son compatibles. El soporte de Microsoft Windows está en progreso. Sin embargo, windi (3rd party software) es una implementación .NET completa del protocolo INDI y está disponible de forma nativa para sistemas Microsoft Windows.
- basado en XML: INDI es pequeño y fácil de analizar. El protocolo INDI se puede anidar dentro de otros elementos XML como RTML para agregar restricciones para la programación y ejecución automática.
- Client agnostic: Los clientes aprenden las propiedades de un dispositivo en particular en tiempo de ejecución mediante la introspección. Esto desacopla la implementación de controlador de dispositivo de los clientes. El controlador de dispositivo puede ser actualizado completamente e independiente de los controladores del cliente.

- Control distribuido: Con la arquitectura cliente/servidor de INDI, puede comunicarse con dispositivos ya sea localmente o de forma remota de forma transparente. INDI acomoda servidores intermedios, broadcasting, y topologías de conexión que van desde punto a punto en un sistema simple a *many to many* entre sistemas de diferente género.
- Ampliamente soportado: INDI es compatible con muchos dispositivos astronómicos incluyendo telescopios, CCD, focos, espectrómetros, y mucho más. Por otra parte, los clientes INDI incluyen suites de software de astronomía populares como KStars, XEphem, SkyCharts, y CDC.

Este software esta liberado bajo la licencia GPL-2.1.

5.5. OROCOS

El OROCOS Toolchain es su principal herramienta para crear aplicaciones de robótica en tiempo real mediante componentes modulares de tiempo de ejecución de software configurable.



Figura 73: Logo del software de control robótico OROCOS

Ofrece:

- Soporte multi-plataforma: Linux, Windows (Visual Studio) y Mac OS-X.
- Extensiones a otros frameworks de robótica: ROS, Rock, Yarp.
- Generadores de código para la transferencia de datos definidos por el usuario entre componentes distribuidos.
- Componentes scriptables y configurables en modo real-time y en modo run-time.
- Registro y notificación de eventos y datos del sistema.

Y se compone de:

- AutoProj: una herramienta para descargar y compilar las librerías necesarias (opcional).
- El Real-Time Toolkit: un componente del framework que permite escribir componentes en C++.
- La biblioteca de componentes OROCOS: los componentes necesarios para iniciar una aplicación e interactuar con ella en tiempo de ejecución.
- OroGen y TypeGen: herramientas para generar código ready-to-compile-and-run a partir de cabeceras existentes o de los archivos de descripción de componentes.

Este software esta liberado bajo la licencia GPL.

5.6. ROS

El Robot Operating System (ROS) es un conjunto de bibliotecas de software y herramientas que ayudan a crear aplicaciones de robots. De los drivers a los algoritmos del estado, y con potentes herramientas de desarrollo, ROS tiene lo que se necesita para los proyectos de robótica. Y todo es de código abierto.



Figura 74: Logo del software de control robótico ROS

Infraestructura de comunicaciones

En el nivel más bajo, ROS ofrece una interfaz de paso de mensajes que proporciona comunicación entre procesos y que comúnmente se conoce como un middleware.

El middleware ROS ofrece estas prestaciones:

- Publicación y suscripción paso de mensajes anónimos.
- Grabación y reproducción de mensajes.
- Petición y respuesta de llamadas a procedimiento remoto.
- Sistema de parámetros distribuidos.

Características específicas para robots

Además de los componentes de middleware del núcleo, ROS proporciona bibliotecas específicas para robots y herramientas. Éstas son sólo algunas de las capacidades que ROS proporciona:

- Definiciones estandares para robots.
- Librería de geometría para robots.
- Lenguaje de descripción de robots.
- Llamadas a procedimiento remoto.
- Diagnóstico.
- Estimación.
- Localización.
- Cartografía.
- Navegación.

Herramientas

Uno de los rasgos más característicos de ROS es el poderoso conjunto de herramientas de desarrollo. Estas herramientas apoyan la introspección, la depuración, el trazado, y visualizar el estado del sistema que se está desarrollando. El mecanismo de publicación y suscripción subyacente permite la introspección de forma espontánea de los datos que fluyen a través del sistema, por lo que es fácil de comprender y depurar problemas a medida que ocurren. Las herramientas de ROS se aprovechan de esta capacidad de introspección a través de una extensa colección de utilidades de línea de comandos y gráficos que simplifican el desarrollo y depuración.

- Herramientas de línea de comandos
- rviz: puede visualizar muchos de los tipos de mensajes comunes previstos en ROS, como escaneos láser, nubes de puntos tridimensionales, imágenes de la cámara, ...
- rqt: Un framework basado en Qt para el desarrollo de interfaces gráficas. Se pueden crear interfaces personalizadas y configurar la amplia biblioteca de plugins en pestañas, en pantalla dividida, y otros diseños.

Este software esta liberado bajo la licencia BSD.

5.7. Conclusiones

Como resumen, hay que decir que:

- ASCOM: Este proyecto, más que un software de control, es un protocolo de abstracción de hardware, además de que sólo funciona en entornos Microsoft Windows, cosa que descarta por completo su uso.
- INDI: Este software es ideal para el control de hardware (es el mismo concepto que ASCOM), y incorpora, como el RTS2, mecanismos para ejecutar scripts aunque sigue sin poder ser una solución valida pues no se puede especificar el workflow de operaciones necesario para el manejo del TJO.
- RTS2, OROCOS y ROS: Tras inspeccionar los códigos, he visto que el coste de añadir el control del telescopio es elevado y no soluciona el principal problema: la toma de decisiones. Actualmente con estos programas se pueden hacer secuencias de órdenes, pero añadir un workflow complejo de operaciones no es viable dado a que es lo mismo que programar un programa externo que le indique las órdenes de lo que tiene que hacer en cada momento.
- TALON: Este es el software que actualmente se está usando y aunque únicamente dispone de entorno gráfico, es viable y rápido de hacer un interfaz para poderlo controlar por línea de comandos.

Además, indicar que estas soluciones no cumplen con algunas de las premisas marcadas en los *Requerimientos del software*, como la de poder hacer evolutivos de forma rápida o ser independiente del hardware que debe controlar. Esto nos lleva a plantear un diseño específico de nuestra necesidad, tal como se verá en el capítulo de *Diseño*.

6. Diseño del nuevo software de control

OpenROCS esta programado como servicios independientes que se comunican entre sí para intercambiar información:

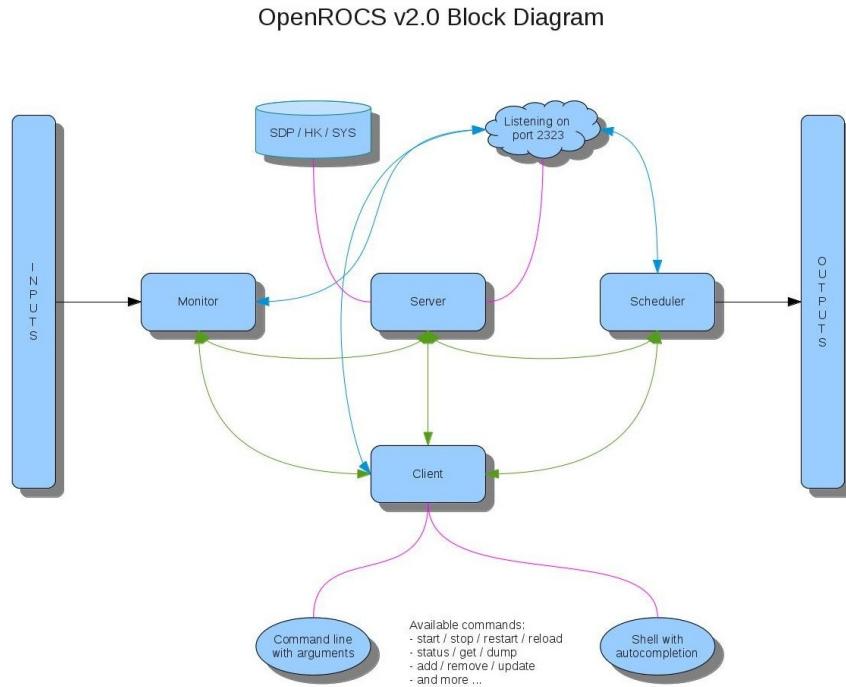


Figura 75: Diagrama de bloques básico de OpenROCS v2.0

En este diagrama se identifican 3 servicios básicos:

- Servidor: que atiende a las solicitudes usando sockets TCP/IP y proporciona almacenamiento remoto con opciones para solicitar los datos utilizando marcas de tiempo totales o incrementales.
- Monitor: la tarea principal de este servicio es mantener actualizado el SDP y HK para ser utilizado por el scheduler.
- Scheduler: permite ejecutar acciones que controlan el telescopio cuando el SDP o HK cambian.

Este diseño da solución al problema, pero ¿que sucede si un servicio falla?. Si no se hace nada correctivo al diseño del diagrama anterior, lo que se tendría es un sistema inestable, porque no se tendría la garantía de que se está ejecutando lo que realmente se espera.

Para añadir un control más exhaustivo de que todos los servicios están funcionando de forma correcta, la solución es añadir unos procesos de watchdog, que garanticen que el sistema está operativo porque todos los servicios involucrados están funcionando tal como se espera. Otra de las cosas que en este diseño se ven latentes es el hecho de que quizás sea interesante tener instancias de OpenROCS ejecutándose en máquinas diferentes, con lo que la comunicación por pipes no sería válida. Para ello, se propone añadir al modelo presentado un servicio adicional que se encargue de intercomunicar instancias de OpenROCS ubicadas en equipos diferentes.

El siguiente diagrama muestra el diseño propuesto para OpenROCS.

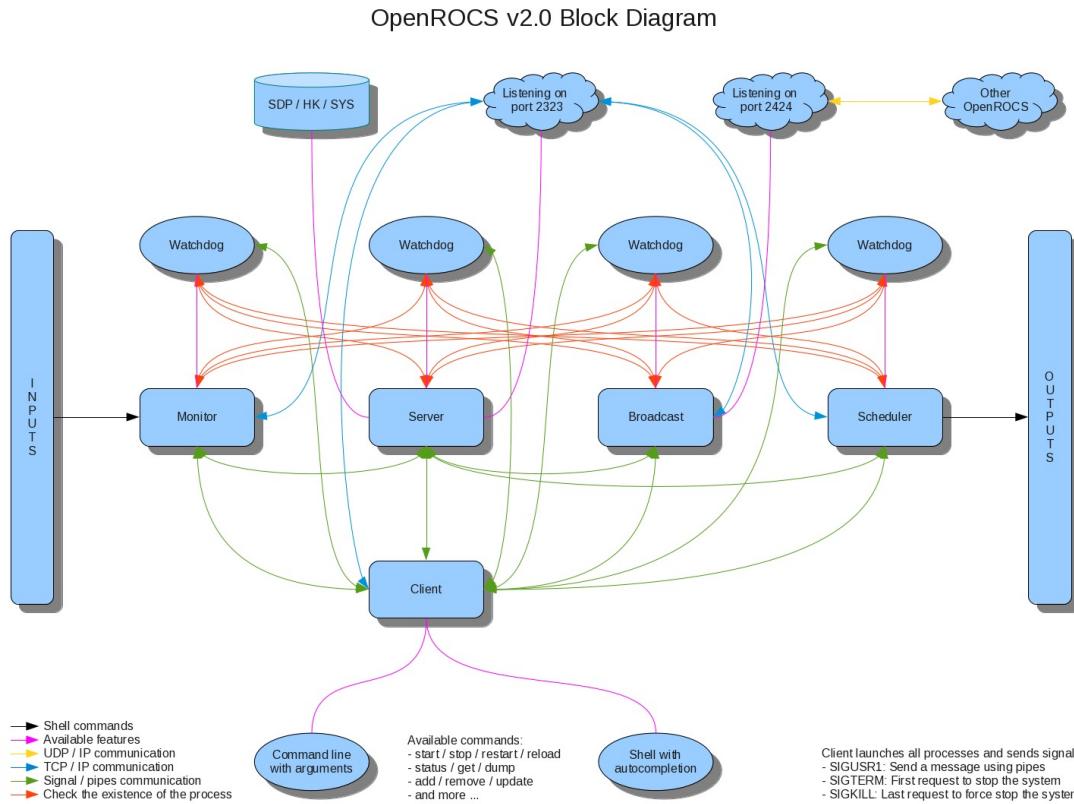


Figura 76: Diagrama de bloques completo de OpenROCS v2.0

El sistema está organizado como:

- 4 servicios principales:
 - Servidor: que atiende a las solicitudes usando sockets TCP/IP y proporciona almacenamiento remoto con opciones para solicitar los datos utilizando marcas de tiempo totales o incrementales.
 - Broadcast: sincroniza múltiples OpenROCS distribuidos en una red mediante el uso del broadcasting para descubrir nuevos OpenROCS en la red.
 - Monitor: la tarea principal de este servicio es mantener actualizado el SDP y HK para ser utilizado por el scheduler.
 - Scheduler: permite ejecutar acciones que controlan el telescopio cuando el SDP o HK cambian.
- Un cliente para gestionar los servicios: permitir el control de los servicios (start/stop/restart/reload) y proporcionar dos modos de uso (por linea de comandos con argumentos o mediante una shell interactiva con autocompletado).
- Un watchdog para cada servicio: en realidad el watchdog estará integrado con el servicio, es decir, no será un proceso adicional, sino que el mismo servicio tendrá una rutina que se ejecutará de forma periódica comprobando que todos los procesos involucrados en OpenROCS están funcionando de forma adecuada y esperando respuesta del resto de watchdogs del resto de servicios.

6.1. Diseño de los módulos

Todos los servicios involucrados en la ejecución de OpenROCS usan porciones de código que comparten prestaciones con el resto, como es el caso del watchdog, el mecanismo de comunicación y la gestión de

procesos. Estas prestaciones están agrupadas para hacer un sistema más robusto que compruebe todo el tiempo el estado del resto del sistema. En otras palabras: el servidor comprueba que el resto de procesos involucrados están funcionando como se espera, el monitor y el scheduler hacen lo mismo. Con este diseño, la redundancia del watchdog aporta un control adicional de que todos los sistemas están trabajando como se espera. Si alguna parte de este sistema fallara, es más razonable que todo se pare en lugar de que continúen ejecutándose servicios pero sin garantizar que la totalidad del sistema esté funcionando tal y como se espera.

6.1.1. El servicio Servidor

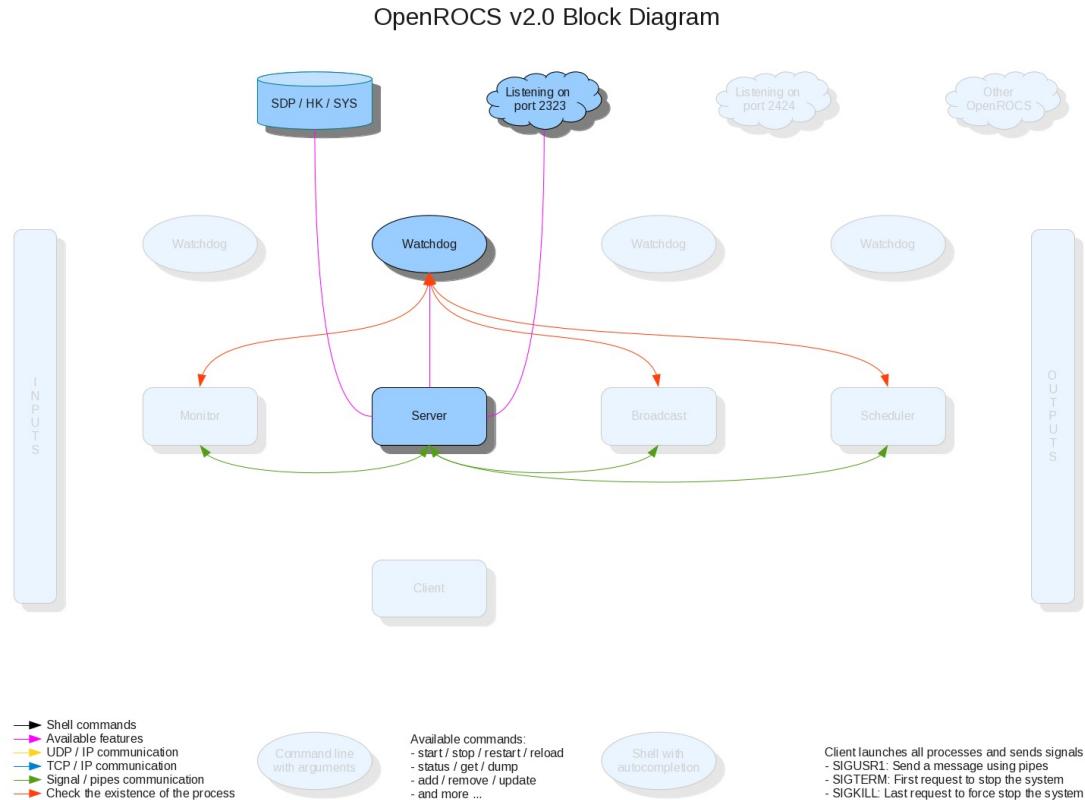


Figura 77: Diagrama de bloques del servicio servidor de OpenROCS v2.0

Este servicio tiene los siguientes objetivos:

- Ejecutar un servidor TCP/IP que atienda a las peticiones del puerto 2323.
- Proveer un sistema de almacenamiento de datos remoto.
- Proveer un lenguaje que gestione las carpetas (stacks), sus variables y sus valores.

La idea principal de este servicio es centralizar toda la información de todo el sistema y permitir a los *clientes* de este servicio poder obtener, crear, modificar y borrar los datos almacenados en los stacks. Existe una gran cantidad de comandos que permiten a los clientes recuperar todas las variables de una pila o sólo recuperar las variables actualizadas desde un timestamp concreto, cosa que permite que el sistema sea más eficiente y evitar la ocupación de ancho de banda de forma innecesaria. Los comandos que el servidor puede entender se explican en la sección [La herramienta Cliente](#).

También, este servicio permite comunicarse directamente con los procesos utilizando señales y tuberías. A partir de los comandos soportados por el servidor, es posible hacer una pausa y continuar la ejecución de un cierto servicio, útil para prevenir problemas de concurrencia cuando, por ejemplo, un scheduler que necesita actualizar la misma variable que un monitor, permitiendo al scheduler detener el monitor, modificar la variable con un acceso exclusivo, y cuando termine la tarea, volver a iniciar el monitor.

6.1.2. El servicio Broadcast

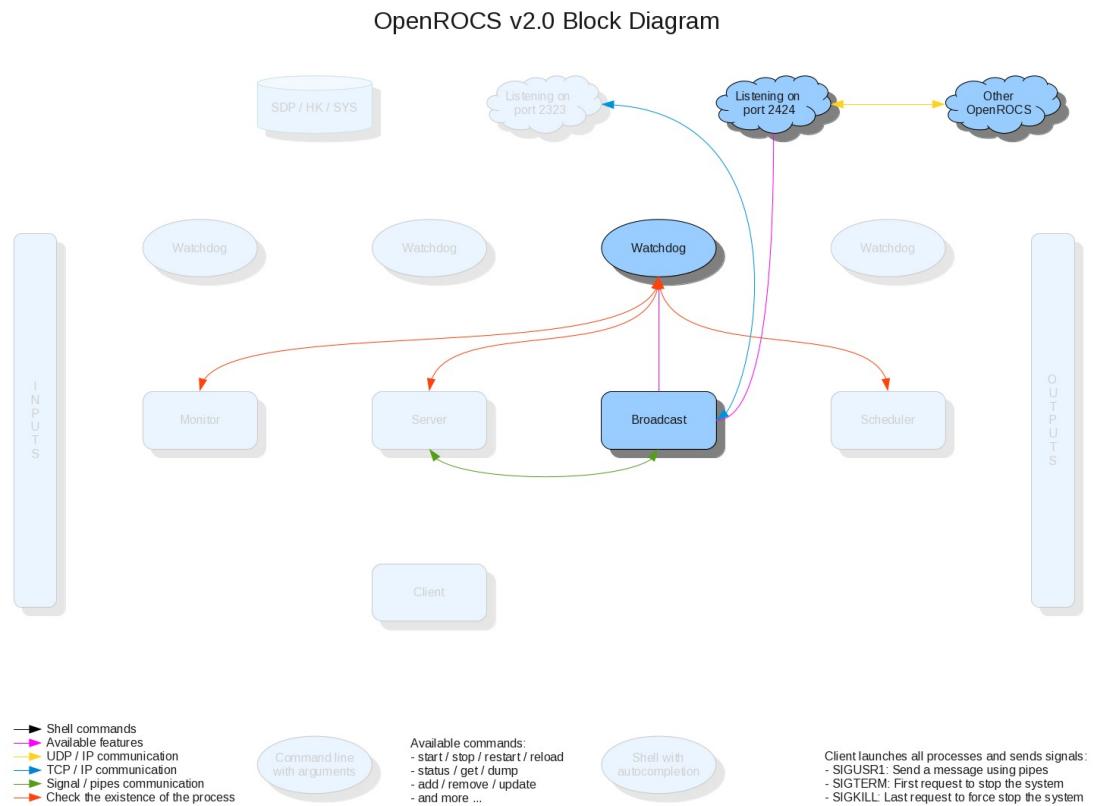


Figura 78: Diagrama de bloques del servicio broadcast de OpenROCS v2.0

Este servicio tiene los siguientes objetivos:

- Ejecutar un servidor UDP/IP en el puerto 2424 para atender las peticiones de broadcasting.
- Enviar paquetes a la dirección de broadcast para descubrir nuevos OpenROCS.
- Actualizar los stacks y variables de los OpenROCS remotos con la información del servidor.

La idea de este servicio es la de permitir a OpenROCS descubrir otras instancias de OpenROCS ejecutándose en la misma red y intercambiar información periódicamente. Esto es útil cuando otros OpenROCS requieren tomar decisiones a partir de las variables de una instancia remota de OpenROCS.

6.1.3. El servicio Monitor

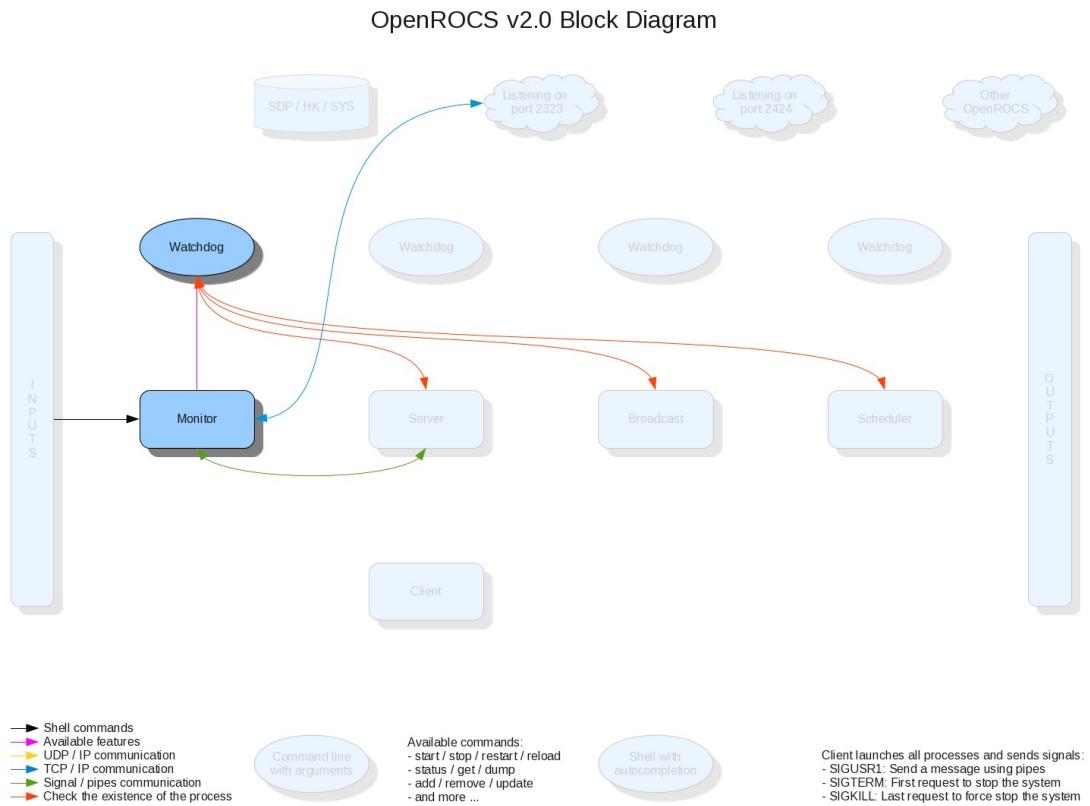


Figura 79: Diagrama de bloques del servicio monitor de OpenROCS v2.0

Este servicio tiene los siguientes objetivos:

- Ejecutar periódicamente las tareas programadas y dependiendo de su salida (stdout y stderr), añadir, eliminar o actualizar con el valor deseado los diferentes stacks (SDP/HK/SYS).
- Configurar tareas que se ejecutarán entre intervalos o a una frecuencia constante: la diferencia entre intervalo y frecuencia es que intervalo es el tiempo entre la finalización y el siguiente inicio de la misma tarea, mientras que frecuencia, es cada x tiempo, indistintamente de lo que pueda tardar.
- Controlar las tareas usando timeouts y programar acciones cuando "ontimeout" es alcanzado.
- Permitir definir estructuras de control a partir del output generado por los comandos. Para hacer esta prestación, se usarán la estructura de lenguaje tipo choose/when/otherwise.
- Comunicación entre el servicio servidor para leer y escribir información (que es el resultado esperado de este servicio).

La idea de este servicio es permitir al sistema la ejecución periódica de tareas y actualizando las variables de ciertos stacks usando las salidas generadas por los comandos usados en esas tareas. Esto es útil, por ejemplo, para monitorizar el estado de las estaciones meteorológicas, los estados de las salidas de las PDUs, monitorizar el estado de ciertos procesos y demonios implicados en el control del telescopio. Cuando el sistema se inicie, todos los schedulers deberán estar parados hasta que todos los monitores hayan ejecutado, como mínimo, una vez todas sus tareas. Esto se hace así para prevenir que los schedulers puedan tomar decisiones erróneas (se

debe entender que los monitores son los encargados de establecer el primer valor para todas las variables del sistema y por ello, es necesario que como mínimo se ejecuten una vez todas sus tareas y así garantizar que los schedulers tomen decisiones a partir de datos correctamente inicializados). El estado inicial de los monitores cuando arranca el sistema es *init* hasta que todas las tareas de cada monitor se hayan ejecutado como mínimo una vez.

6.1.4. El servicio Scheduler

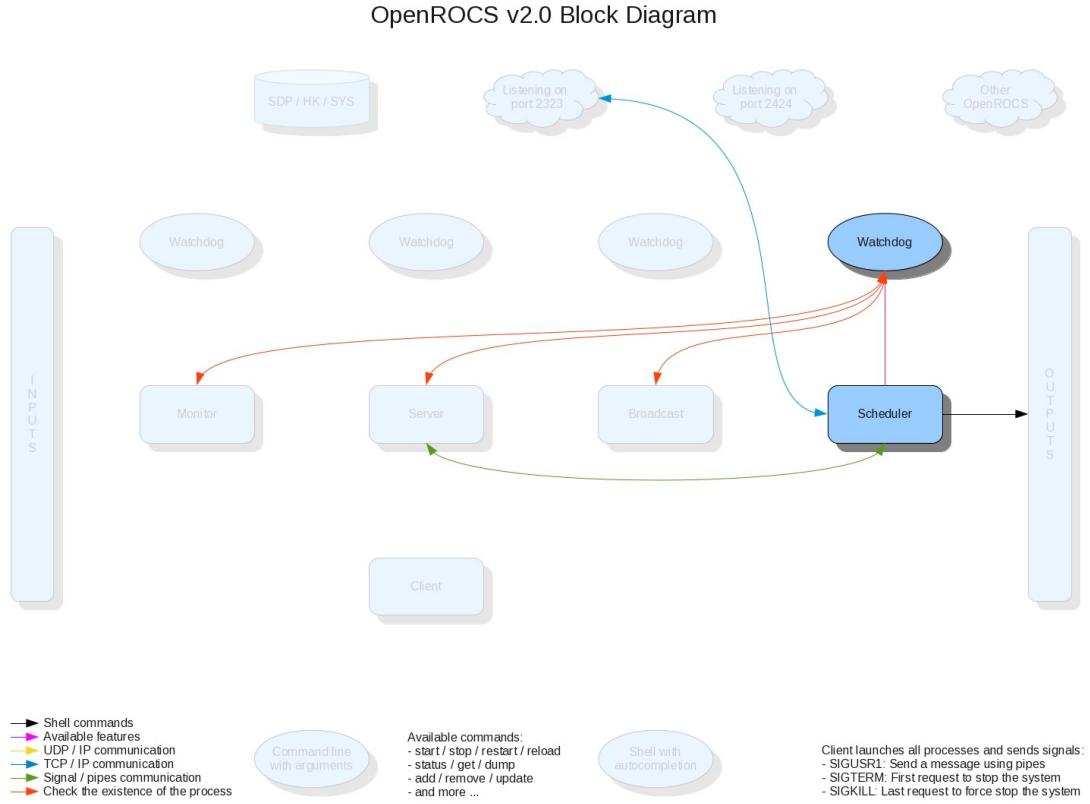


Figura 80: Diagrama de bloques del servicio scheduler de OpenROCS v2.0

Este servicio tiene los siguientes objetivos:

- Ofrecer un sistema de monitorización de variables que será usada para ejecutar iteraciones del scheduler cuando ciertas variables del SDP/HK/SYS cambien (hay que recordar que el servicio monitor ejecuta sus tareas de forma periódica sin necesidad de que nada cambie).
- Este servicio puede definir mas de una instancia de ejecución del scheduler con sus variables (el concepto de hash que se explicará más adelante).
- Obtener y establecer datos desde y hacia el servicio servidor. Para que este proceso sea más eficiente cuando se deba obtener la información del servidor, se usará la sincronización incremental, es decir, que sólo se obtendrán los stacks y variables que hayan sido añadidas, actualizadas o borradas usando una marca de timestamp como control.

La idea de este servicio es la de permitir al sistema la toma de decisiones dependiendo del valor de las variables de los stacks. Cada scheduler debe definir el *hash variables*, que es la lista de variables que pueden

lanzar la ejecución de una iteración del scheduler. Como está explicado en el *servicio monitor*, el scheduler debe esperar hasta que todos los monitores establecen las variables necesarias para prevenir ejecuciones erróneas con datos no iniciados. Por esta razón, el estado inicial de los schedulers será *stop* mientras espera a que todos los monitores pasen del estado *init* al estado *start*.

6.1.5. La herramienta Cliente

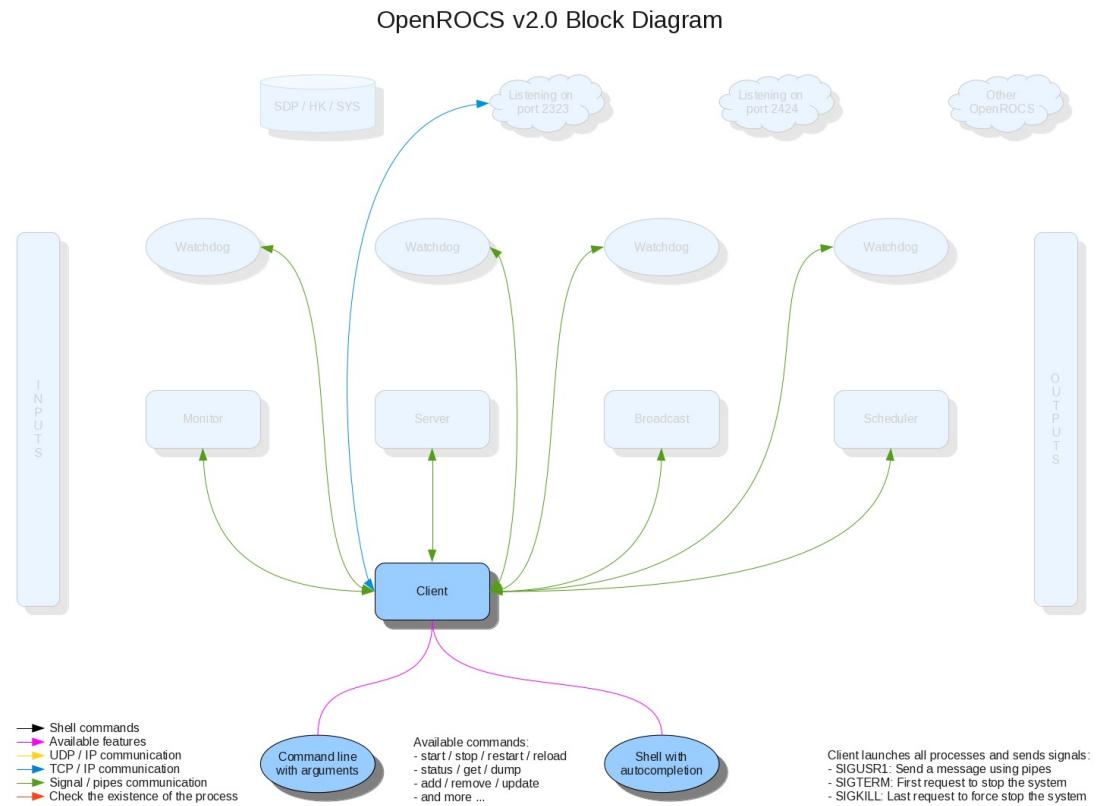


Figura 81: Diagrama de bloques de la herramienta cliente de OpenROCS v2.0

La herramienta cliente esta diseñada para permitir a los operadores del telescopio monitorizar y modificar el comportamiento de los monitores y schedulers mediante las opciones que permiten añadir, modificar y borrar datos de los stacks definidos (SDP/HK/SYS). Este cliente debe permitir trabajar en dos modos:

- Mediante linea de comandos con argumentos: para permitir comandos de OpenROCS con argumentos que se ejecuten en modo síncrono (es decir, con bloqueo hasta su finalización)
 - Modo interactivo de shell con autocompletado: para permitir la interacción humana usando un shell típico con autocompletado (como lo hace el bash, por ejemplo).

```

Terminal - sanz@localhost:~
Archivo Editar Ver Terminal Pestañas Ayuda
[sanz@localhost ~]$ orocs
[OpenROCS v2.0]
orocs# start
Created new child server[12138@mylaptop] Server up, listening on port 2323
Created new child broadcast[12139@mylaptop] Broadcast up, listening on port 2424
Created new child monitor#pingpong[12140@mylaptop] Monitor up
Created new child scheduler#ping[12141@mylaptop] Scheduler up
Created new child scheduler#pong[12142@mylaptop] Scheduler up
Watchdog started in server[12138@mylaptop]
Watchdog started in broadcast[12139@mylaptop]
Watchdog started in monitor#pingpong[12140@mylaptop]
Watchdog started in scheduler#ping[12141@mylaptop]
Watchdog started in scheduler#pong[12142@mylaptop]
Sending start to broadcast[12139@mylaptop]
Sending start to monitor#pingpong[12140@mylaptop]
Server running
orocs# 

```

Figura 82: Herramienta cliente de OpenROCS v2.0

Los dos modos tienen las mismas opciones y comandos. La idea de disponer de estos modos de ejecución es, por ejemplo, para integrarlo con otro software o shell scripts y proporcionar una herramienta que permita interaccionar con el sistema de forma fácil y rápida.

Los comandos sin parámetros son:

- shell: abrir una shell interactiva (por defecto, si se ejecuta sin pasar argumentos).
- help: muestra la pantalla de ayuda.
- start: lanza la ejecución de todos los servicios.
- restart: reinicia todos los servicios.
- reload: igual que el *restart*, pero sin perder datos.
- stop: para todos los servicios.
- status: chequear el estado del servidor.
- dump: mostrar una lista de los stacks con sus variables y valores para una rápida interpretación.
- check: ejecuta unos tests de validación del sistema rápidos.
- history: muestra la lista de comandos ejecutados (sólo disponible en el modo de shell interactivo).

Los comandos que requieren parámetros son:

- get: retorna una lista con todos los stacks presentes.
- get STACK: retorna una lista con todas las variables del STACK.
- get STACK KEY: retorna el valor de la variable KEY del STACK.
- get TIMESTAMP: retorna una lista con los stacks que se han modificado desde el TIMESTAMP.
- get STACK TIMESTAMP: retorna una lista con los datos modificados del STACK desde el TIMESTAMP.
- add/create STACK: crea el STACK si no existe.
- add/create STACK KEY: añade la variable KEY al STACK.
- add/create STACK KEY=VALUE: añade la variable KEY con su valor VALUE al STACK.
- update/set STACK KEY=VALUE: actualiza en el STACK la variable KEY con el nuevo valor VALUE.

- remove/delete STACK: borra el STACK sólo si existe y esta vacio.
- remove/delete STACK KEY: borra la variable KEY del STACK si existe.
- stop SERVICE: pausa el SERVICE.
- status SERVICE: chequea el estado del SERVICE.
- start SERVICE: continua el SERVICE.

Notas:

- Los comandos start y stop, estan sólo disponibles desde la linea de comandos (usando argumentos o usando la shell interactiva). Si intenta enviar estos comandos al servidor directamente, la respuesta será *permission denied*.
- Otros comandos como shell, help, dump, check y history, están sólo disponibles desde el cliente orocs. Si intenta enviar estos comandos directamente al servidor, la respuesta será *unknown command*.

6.2. Interfaces del sistema

OpenROCS, usa varias técnicas para comunicar procesos (Servidor, Broadcast, Monitor, Scheduler y Cliente) y acceder al servicio de almacenamiento remoto (Servidor) para establecer y obtener datos del SDP/HK.

6.2.1. Señales y tuberías (pipes)

Esta técnica de comunicación es usada de tres formas:

- Desde el comando *start* y es usado para comunicar al cliente con los procesos creados y obtener el output generado (el mensaje de la consola).
- Por el sistema de watchdog que envía la lista de todos los procesos a todos los procesos y chequea que obtiene respuesta.
- Por el servidor para cambiar el estado del resto de procesos.

6.2.2. Sockets TCP/IP

Esta técnica de comunicación es la forma más genérica usada por el resto de sistemas implicados en OpenROCS:

- El servicio servidor, ejecuta un servidor para atender las peticiones del puerto 2323 y responder con la respuesta esperada.
- El cliente envía y recibe datos desde y hacia el SDP/HK/SYS y CHLD (procesos hijos lanzados por OpenROCS como servicios).
- El monitor envía (y también puede recibir) todos los resultados de las tareas al servicio de almacenamiento remoto (SDP/HK/SYS).
- El scheduler recibe y envía información de SDP/HK/SYS.
- El broadcast usa este canal para actualizar los stacks con los datos que provienen de otros OpenROCS.

6.2.3. Sockets UDP/IP

Esta técnica de comunicación es usada por el servicio de broadcast:

- El broadcast ejecuta un servidor para atender las peticiones del puerto 2424 y responder con la respuesta esperada.
- También envía periódicamente paquetes a la dirección de broadcast para descubrir otros OpenROCS que se están ejecutando en la misma red.

6.2.4. Comandos shell

En los objetivos de OpenROCS, ya se ha explicado que el diseño pretende definir un modelo abstracto para ser independiente del hardware y del software del telescopio. Para hacer esta capa de abstracción, OpenROCS no implementa un código específico para conectarse, comunicarse y actuar con el telescopio y otros elementos como por ejemplo, los sistemas de meteorología. La única manera de comunicarse con los sistemas externos para recibir datos y para actuar sobre las salidas es la línea de comandos (shell).

Este concepto permite a OpenROCS usar herramientas externas como:

- SNMP para obtener o establecer parámetros a dispositivos (PDUs, UPSs, estaciones meteorológicas y muchos otros equipos que pueden comunicarse usando este protocolo estándar y extendido).
- Clientes INDI para comunicarse y controlar el telescopio y las cámaras, por ejemplo.
- Otros scripts que ofrecen similares prestaciones que los comandos comentados anteriormente, por ejemplo, TALON cuando escribe y lee directamente hacia y desde fifos (que son pipes con nombre).

Como se puede ver, OpenROCS ofrece varios métodos útiles para obtener y establecer datos con el resto de sistemas. Todos los comandos pueden usar el stdout y el stderr para generar la salida esperada. OpenROCS permite evaluar estas salidas (es decir, los canales que todos los procesos de sistemas UNIX disponen) usando un lenguaje intuitivo (PHP por defecto).

6.3. Configuración del sistema

El sistema usa XML para definir todas las variables y estructuras de datos que necesita y por ello, es importante usar correctamente la sintaxis del formato XML. En algunos casos, será necesario usar "trucos" para cumplir la especificación de este formato:

- Si el valor de un nodo contiene los caracteres "&" o "<", el valor de este nodo debe encerrarse dentro de un bloque de tipo CDATA:

```
1  <eval><! [CDATA[$OUTPUT >=-200 && $OUTPUT <=200]]></eval>
```

Código 1: Ejemplo correcto de uso de CDATA en un nodo XML

- Sin la cláusula CDATA, el parser de XML generará un error cuando lea el nodo:

```
1  <eval>$OUTPUT >=-200 && $OUTPUT <=200</eval>
```

Código 2: Ejemplo erróneo de uso sin CDATA en un nodo XML

- Este es el formato de XML que todos los ficheros deben de cumplir para que puedan ser parseados correctamente:

```

1  <?xml version="1.0" encoding="UTF-8" ?>
2  <root>
3    *
4    *
5    *
6  </root>
```

Código 3: Ejemplo de formato de un fichero XML

6.3.1. El fichero config.xml

Este fichero es usado por todos los servicios y el cliente. Define la configuración del servidor y los stacks por defecto, los parámetros de depuración, las prestaciones de la shell y establece algunos parámetros de configuración (php.ini y variables de entorno).

Para más información, véase el Anexo 12.1:

- Anexo 12.1.1: [El nodo <server>](#)
- Anexo 12.1.2: [El nodo <broadcast>](#)
- Anexo 12.1.3: [El nodo <debug>](#)
- Anexo 12.1.4: [El nodo <shell>](#)
- Anexo 12.1.5: [El nodo <timeout>](#)
- Anexo 12.1.6: [El nodo <polling>](#)
- Anexo 12.1.7: [El nodo <retries>](#)
- Anexo 12.1.8: [El nodo <ini set>](#)
- Anexo 12.1.9: [El nodo <putenv>](#)

En el caso de la implementación del TJO, se ha usado el siguiente fichero de configuración:

- Anexo 12.10.1: [config.xml](#)

6.3.2. El fichero variables.xml

Este fichero permite definir variables que luego se podrán usar en la definición del resto de ficheros XML (como por ejemplo, el monitor.xml y el scheduler.xml). El árbol, permite tener múltiples niveles y la variable resultante es la concatenación de todos los nombres de los nodos usando el carácter "understand" como separador entre nodos. Por ejemplo, para usar el valor del nodo IP => SERVER, debe usarse la variable \$IP_SERVER.

```

1  <variables>
2    <IP>
3      <SERVER>192.168.0.10</SERVER>
4      <SENSORS>192.168.0.11</SENSORS>
5      <TELESCOPE>192.168.0.12</TELESCOPE>
6      <DOME>192.168.0.13</DOME>
7      <CAMERA>192.168.0.14</CAMERA>
8      <POWER>192.168.0.15</POWER>
```

```

9      </IP>
10     <INTERVAL>10</INTERVAL>
11     <PDU>
12       <OID>.1.3.6.1.4.1.318.1.1.12.3.3.1.1.4</OID>
13       <ON>1</ON>
14       <OFF>2</OFF>
15     </PDU>
16   </variables>

```

Código 4: Formato del fichero variables.xml

En el caso de la implementación del TJO, se ha usado el siguiente fichero de variables:

- Anexo 12.10.2: [variables.xml](#)

6.3.3. El fichero monitor.xml

Este fichero define las tareas ejecutadas por cada monitor. El número máximo de tareas dependerá de la memoria y cpu de cada computador que ejecute este monitor. Por definición, se puede entender que el número máximo de tareas no esta limitado. El siguiente ejemplo, muestra el esquema de configuración del monitor.

```

1  <monitor>
2    <name>mymonitor1</name>
3    <task>
4      <interval>10</interval>
5      *
6      *   Process nodes
7      *
8    </task>
9    <task>
10      <frequency>10</frequency>
11      *
12      *   Process nodes
13      *
14    </task>
15    <task>
16      <interval>10</interval>
17      *
18      *   Process nodes
19      *
20    </task>
21  </monitor>
22 <monitor>
23   <name>mymonitor2</name>
24   <task>
25     <frequency>10</frequency>
26     *
27     *   Process nodes
28     *
29   </task>
30   <task>
31     <interval>10</interval>
32     *
33     *   Process nodes
34     *
35   </task>
36   <task>
37     <frequency>10</frequency>
38     *
39     *   Process nodes
40     *
41   </task>
42 </monitor>

```

Código 5: Formato del fichero monitor.xml

Notas:

- <interval> o <frequency>: El disparador (trigger) usado para ejecutar cada tarea.
 - <interval>: define el tiempo transcurrido entre el fin de la ejecución y el inicio de la siguiente ejecución. Por ejemplo, si un comando de shell consume 1 segundo y se define un intervalo de 10 segundos, la ejecución real del comando sera con una frecuencia de cada 11 segundos.
 - <frequency>: Permite indicar una frecuencia precisa de ejecución. Esta forma de definir el intervalo es independiente del tiempo consumido por al ejecución de la shell. Para un uso más seguro de esta prestación, es recomendable usar el tag <timeout> para evitar el solapamiento de ejecuciones.
- Se pueden agrupar tareas usando el nodo <monitor>. Cada nodo <monitor> es ejecutado como un proceso separado que puede ser suspendido y reanudado de forma independiente.
- Se puede especificar el nodo <name> en cada <monitor>. Si el nodo <name> no existe, se usará un número para identificarlo.
- El contenido de cada <task> debe cumplir la especificación de los nodos de proceso.

En el caso de la implementación del TJO, se ha usado el siguiente fichero para la configuración de los monitores:

- Anexo 12.10.3: [monitor.xml](#)

Y este fichero XML ejecuta las acciones de los siguientes ficheros:

- Anexo 12.10.5: [tjo.xml](#)
- Anexo 12.10.6: [dome.xml](#)
- Anexo 12.10.7: [meia.xml](#)
- Anexo 12.10.8: [meteo.xml](#)
- Anexo 12.10.9: [telrun.xml](#)

6.3.4. El fichero scheduler.xml

Este fichero define las tareas ejecutadas por los schedulers. Como se podrá observar, el scheduler.xml tiene las mismas opciones que el fichero monitor.xml y la única diferencia es que el scheduler.xml debe definir para cada scheduler un nodo <hash> con las variables que se usarán como disparador de la ejecución de cada scheduler (trigger).

```

1  <scheduler>
2    <name>myscheduler1</name>
3    <hash>
4      <variable>myvariable1</variable>
5      <variable>myvariable2</variable>
6      <variable>myvariable3</variable>
7    </hash>
8    *
9    *   Process nodes
10   *
11  </scheduler>
12  <scheduler>
13    <name>myscheduler2</name>
14    <hash>
15      <variable>myvariable4</variable>
16      <variable>myvariable5</variable>
17      <variable>myvariable6</variable>
18    </hash>
19    *
20    *   Process nodes
21

```

Código 6: Formato del fichero scheduler.xml

Donde:

- El nodo <scheduler> define una tarea del scheduler.
- Cada nodo <scheduler> requiere del nodo <hash> para definir las variables que serán usadas para calcular el hash (usado como disparador).
- Cada nodo <scheduler> será ejecutado en un proceso independiente que puede ser suspendido y reanudado de forma independiente.
- Se puede especificar el nodo <name> en cada <monitor>. Si el nodo <name> no existe, se usará un número para identificarlo.
- El contenido de cada <action> y <scheduler> debe cumplir la especificación de los nodos de proceso.

En el caso de la implementación del TJO, se ha usado el siguiente fichero para la configuración de los schedulers:

- Anexo 12.10.4: [scheduler.xml](#)

Y este fichero XML ejecuta las acciones de los siguientes ficheros:

- Anexo 12.10.5: [tjo.xml](#)
- Anexo 12.10.6: [dome.xml](#)
- Anexo 12.10.7: [meia.xml](#)
- Anexo 12.10.8: [meteo.xml](#)
- Anexo 12.10.9: [telrun.xml](#)

6.4. Los nodos de proceso

Los nodos de proceso, permiten especificar comandos, evaluaciones y control de flujo que debe usarse en los monitores y schedulers para conseguir el objetivo. En realidad, estos nodos definen el lenguaje real usado por OpenROCS para programar los que realmente se requiere. Con estos nodos, se puede programar un workflow que ejecute comandos, procese la salida y ejecute otros comandos dependiendo de la evaluación que se deseé. El control de flujo se hará mediante nodos de tipo <choose>.

Para más información, véase el anexo 12.2:

- Anexo 12.2.1: [El nodo <action>](#)
- Anexo 12.2.2: [El nodo <shell>](#)
- Anexo 12.2.3: [El nodo <php>](#)
- Anexo 12.2.4: [El nodo <choose>](#)
- Anexo 12.2.5: [El nodo <send>](#)
- Anexo 12.2.6: [El nodo <log>](#)

7. Tareas previas a la implementación

En este capítulo se explica la necesidad de realizar cambios en algunas partes del sistema existente para permitir que el nuevo diseño controle todo los elementos del TJO y conseguir así la completa robotización del sistema.

Para ello, ha sido necesario realizar cambios en algunos elementos que forman parte el sistema de control del TJO:

- Sistema de control del telescopio:
 - TALON: este software actualmente se usa mediante interfaz gráfico. Para conseguir el control robótico, ha sido necesario poder ejecutar las prestaciones del software de control actual mediante linea de comandos. Para ello, se ha desarrollado un pequeño programa que actuará como cliente de talon y permite hacer todas las operaciones del telescopio sin necesidad de interfaz gráfico.
 - Sistemas de meteorología:
 - Previstorm: Este hardware estaba pendiente de ser añadido al sistema de monitorización del entorno. Para poder obtener los datos de este hardware, se ha desarrollado un pequeño programa en C que permite leer los datos para su posterior publicación mediante SNMP.
 - EM-Davis: Este sistema es una estación meteorológica y funciona mediante el software OpenROCS v1.0, que implementa la lectura de los datos de esta estación meteorológica y los publica en un servicio tipo telnet en el puerto 6666. Para el nuevo diseño, se ha desarrollado un pequeño programa en C que permite leer los datos y posteriormente publicarlos mediante un servidor SNMP.
 - EM-SMC: Este sistema es un datalogger del Servei Meteorològic de Catalunya y funcionaba con un equipo con Microsoft Windows y una aplicación privativa que el propio SMC instaló para poder obtener los datos de la propia estación meteorológica. Para aplicar el modelo deseado, se ha reemplazado este programa por un paquete de software libre llamado PBCdIComm que permite obtener los datos de la estación desde un sistema GNU/Linux.
 - EM-TFRM: Este sistema es una estación meteorológica Vaisala que pertenece al telescopio TFRM y se puede obtener los datos usando un cliente INDI (que es muy parecido al protocolo SNMP).
 - Sensor de lluvia: El sensor de lluvia únicamente consiste en un cable de 2 hilos que activa la señal TTL cuando detecta lluvia y transcurrido un tiempo, desactiva la señal TTL. Se puede obtener el valor de la linea mediante un puerto RS-232.
 - Sensor de nubes Boltwood: Este hardware estaba pendiente de ser añadido al sistema de monitorización del entorno. Para poder obtener los datos de este hardware, se ha usado el software que el propio fabricante tiene disponible para sistemas GNU/Linux.

7.1. Sistema de control del telescopio

Para controlar el telescopio usando talon desde linea de comandos, se ha implementado un script (anexo 12.3.1: [talon_fifo](#)) que permite escribir en las fifos de talon y leer las respuestas de forma síncrona. Este script permite múltiples opciones como:

- Escribir en una fifo sin esperar respuesta
- Ídem que el anterior esperando una o múltiples respuestas
- Ídem que el anterior especificando un timeout

Este script, usa un diccionario de comandos (anexo 12.3.2: [talon_alias](#)) que permite a los operadores configurar y modificar el comportamiento de este script según las necesidades.

Con estas opciones, el programa talon pasa a ser valido para el diseño presentado anteriormente.

7.2. Sistemas de meteorología

Para llevar a cabo el proceso de automatización de la meteorología, se han desarrollado pequeños programas en C y PHP que permiten acceder a cada dispositivo de forma sencilla. Estos programas desarrollados en C permiten además, detectar cuando se pierde la comunicación con el dispositivo para volver a intentar establecerla. Este tipo de funcionalidad es útil pues en ocasiones, se pueden producir cortes en las comunicaciones por tareas de mantenimiento o porque se desactiven ciertas partes del sistema por motivos de cortes de suministro eléctrico.

Para poner en marcha esta parte de la migración, se han desarrollado los siguientes códigos:

- Anexo 12.4.1: [previstorm.c](#)
- Anexo 12.4.2: [davis.c](#)
- Anexo 12.4.3: [emsmc.php](#)
- Anexo 12.4.4: [EMSMC.xml](#)
- Anexo 12.4.5: [vaisala.php](#)
- Anexo 12.4.6: [raindetect.c](#)
- Anexo 12.4.7: [cloudsensor.c](#)

Notas:

- El anexo 12.4.3 es el script que se ha desarrollado para ejecutar el programa PBCdlComm.
- El anexo 12.4.4 es el fichero de configuración del programa PBCdlComm.

También se ha desarrollado otro módulo que permite obtener la posición del sol desde el OAdM para poder determinar cuando debe iniciarse el proceso de observación y cuando debe finalizar:

- Anexo 12.4.8: [sunpos.py](#)

Y toda la información que estos procesos generan, se publica mediante un servidor snmpd cuyo fichero de configuración es:

- Anexo 12.4.9: [snmpd.conf](#)

8. Implementación

8.1. Lenguaje de programación

El lenguaje utilizado para programar OpenROCS ha sido *PHP*, un lenguaje de scripting generalizado con una gran cantidad de años de pruebas y con el apoyo de la comunidad. Es similar a Python o Perl, pero utiliza una sintaxis muy parecida al lenguaje *C*.

Como un ejemplo del potencial de PHP, OpenROCS utiliza características como:

- `pcntl_fork` y `pcntl_signal`: para permitir a OpenROCS iniciar scripts en segundo plano y comunicarse como si fuera un programa escrito en *C*.
- `register_tick_function` y `unregister_tick_function`: para permitir a OpenROCS programar una tarea periódica (se utiliza como sistema de control watchdog).
- `socket_create`, `socket_bind`, `socket_listen`, `socket_select` y `socket_accept` (utilizado para publicar un servicio de red).
- `register_shutdown_function`: para permitir a OpenROCS programar una tarea de apagado controlada (utilizado para cerrar sockets y ejecutar el recolector de basura).
- `set_error_handler` y `set_exception_handler`: para permitir a OpenROCS programar las rutinas de error que aportan el control necesario en caso de fallo del programa.

También, *PHP* proporciona una gran cantidad de funcionalidades para trabajar con estructuras como árboles y matrices, analizar cadenas y archivos XML, para usar variables por referencia, la prestación de la variable `variable` y más cosas que OpenROCS explota para llevar a cabo todas las tareas necesarias.

8.2. Organización del código fuente

El código fuente utiliza la siguiente estructura de directorios:

- `root`: Este directorio contiene todo el software necesario para ejecutar OpenROCS y el programa principal que se utiliza en todas las solicitudes (el script principal de OpenROCS, `orocs`)
 - `xml`: Aquí, el software intenta cargar la configuración para cada servicio.
 - `HOSTNAME`: si existe un directorio con el mismo nombre que el nombre de host del equipo que está ejecutando OpenROCS, se usará esta dirección para buscar los archivos necesarios. En caso de no encontrarlos en esta ruta, lo buscará en el directorio `xml` directamente.
 - `php`: Este directorio contiene todos los scripts utilizados para implementar OpenROCS.
 - `action`: Contiene las secuencias de comandos específicos que ejecutan algunas acciones como `start`, `restart`, `reload` and `stop`, `help` y `shell`, ...
 - `log`: Este directorio es utilizado por OpenROCS para almacenar el registro de actividad (logs) que se utilizan para comprobar el correcto funcionamiento y para propósitos de depuración.

8.3. Programación del diseño elegido

OpenROCS sólo tiene un mecanismo de acceso y ejecución: el comando `orocs` (anexo 12.5.1: `orocs`).

Este comando es el empleado para lanzar todos los procesos y para comunicarse con los mismos. Como se puede ver en el anexo, el código está organizado de una manera fácil e intuitiva de mantener.

El workflow del script principal es el siguiente:

- Cambiar al directorio que contiene el script que se está ejecutando. Esto es útil para evitar errores a la hora de acceder a archivos de configuración y otros scripts.
- Cargar las librerías empleadas del proyecto SaltOS
- Cargar las librerías del propio proyecto
- Programar el controlador de errores
- Activar el controlador de ticks
- Iniciar el recolector de basura
- Cargar, validar y establecer la configuración del sistema
- Ejecutar la acción requerida

8.4. Funciones heredadas del proyecto SaltOS

Del proyecto SaltOS, se han heredado las funciones que sirven de base para desarrollar OpenROCS v2.0. Estas funciones se enmarcan básicamente en el procesado de ficheros XML, control de errores y control de semáforos.

Para más información, véase el anexo 12.6:

- Anexo 12.6.1: [array2xml.php](#)
- Anexo 12.6.2: [saltos.php](#)
- Anexo 12.6.3: [xml2array.php](#)

8.5. Funciones y servicios de OpenROCS

Para la programación de OpenROCS, se ha empleado una metodología muy típica: agrupar en un mismo fichero todas las funciones que aportan una funcionalidad concreta. De esta manera, cada fichero de código fuente aporta un conjunto de funciones que a su vez, añaden una funcionalidad determinada a OpenROCS. En esta sección, hay ficheros que implementan una funcionalidad concreta, como es el caso de los ficheros *args.php*, *broadcast.php*, *monitor.php*, *scheduler.php* y *server.php*. El resto de ficheros contienen las funciones que serán usadas por los ficheros anteriores.

Para más información, véase el anexo 12.7:

- Anexo 12.7.1: [args.php](#)
- Anexo 12.7.2: [broadcast.php](#)
- Anexo 12.7.3: [checks.php](#)
- Anexo 12.7.4: [child.php](#)
- Anexo 12.7.5: [comm.php](#)
- Anexo 12.7.6: [define.php](#)
- Anexo 12.7.7: [functions.php](#)
- Anexo 12.7.8: [gc.php](#)
- Anexo 12.7.9: [monitor.php](#)

- Anexo 12.7.10: [pipes.php](#)
- Anexo 12.7.11: [process.php](#)
- Anexo 12.7.12: [scheduler.php](#)
- Anexo 12.7.13: [server.php](#)
- Anexo 12.7.14: [signals.php](#)
- Anexo 12.7.15: [stack.php](#)

8.6. Acciones de OpenROCS

A continuación se explicarán las funcionalidades de cada fichero de tipo action.

Para más información, véase el anexo 12.8:

- Anexo 12.8.1: [crontab.php](#)
- Anexo 12.8.2: [help.php](#)
- Anexo 12.8.3: [reload.php](#)
- Anexo 12.8.4: [restart.php](#)
- Anexo 12.8.5: [shell.php](#)
- Anexo 12.8.6: [start.php](#)
- Anexo 12.8.7: [start0.php](#)
- Anexo 12.8.8: [start1.php](#)
- Anexo 12.8.9: [start2.php](#)
- Anexo 12.8.10: [stop.php](#)
- Anexo 12.8.11: [stop1.php](#)
- Anexo 12.8.12: [stop2.php](#)

8.7. Licencia usada

OpenROCS v2.0 esta liberado bajo la licencia GPL-3.0. Para ello, se ha tenido en cuenta que todas las librerías y fragmentos de código empleado sea compatible con esta licencia. Como nota, decir que se han usado porciones de código del proyecto SaltOS, que también esta publicado bajo la licencia GPL-3.0, y que aporta el parser de XML, la gestión de errores y la gestión de semáforos, de entre otras muchas más funcionalidades.

Para que todo esto quede reflejado en el código fuente, se han añadido a todos los ficheros las cabeceras correspondientes:

- Anexo 12.9.1: [Cabecera OpenROCS v2.0](#)
- Anexo 12.9.2: [Cabecera SaltOS 3.1](#)

8.8. Ficheros XML usados en el TJO

A continuación se muestran los ficheros XML usados en el TJO:

- Anexo 12.10.1: [config.xml](#)
- Anexo 12.10.2: [variables.xml](#)
- Anexo 12.10.3: [monitor.xml](#)
- Anexo 12.10.4: [scheduler.xml](#)
- Anexo 12.10.5: [tjo.xml](#)
- Anexo 12.10.6: [dome.xml](#)
- Anexo 12.10.7: [meia.xml](#)
- Anexo 12.10.8: [meteo.xml](#)
- Anexo 12.10.9: [telrun.xml](#)

9. Resultados

Actualmente, el TJO dispone de un sistema de control que ha aportado mejoras en varios aspectos:

- Estabilidad: OpenROCS, lleva funcionando varios años y, excepto en las fases de pruebas, no ha fallado nunca.
- Recuperación ante errores: OpenROCS, no implementa el control de hardware en si mismo, sino que usa programas externos para realizar estas tareas. La monitorización que hace OpenROCS de la ejecución de estos procesos externos, ha demostrado ser robusto y proporcionar el feedback necesario para que el sistema se recupere de errores externos.
- Escalabilidad: OpenROCS ha demostrado que es capaz de adaptarse a las nuevas necesidades, gracias a su lenguaje de especificación. Esto ha permitido añadir, reemplazar y quitar elementos al sistema de forma fácil y rápida manteniendo un sistema que en ningún momento ha dejado de funcionar.
- Adaptabilidad: OpenROCS ha sido portado a otro telescopio (el SuperWASP Qatar Telescope) de forma rápida. proporcionando un sistema de control estable y que facilita el trabajo de los usuarios que tienen que interaccionar con él.

A continuación se muestra una tabla con las observaciones realizadas desde julio del año 2010 hasta finales del año 2014 con el Telescopio Joan Oró. Como se puede ver, a partir del año 2012 aumento el tiempo observado y el numero de imágenes obtenidas, cosa que no sólo se debe al buen funcionamiento y estabilidad del nuevo sistema de control OpenROCS v2.0, sino también al hecho de que cada vez este telescopio es más conocido por la comunidad científica y por lo tanto, hay más demanda de uso.

Año	Noches	Observadas	Frac. noches	Imágenes	Tiempo
2010	184	11	5.98 %	2094	32.9455
2011	365	68	18.63 %	3889	93.9017
2012	366	89	24.32 %	8832	193.0774
2013	365	111	30.41 %	11673	344.7978
2014	365	190	52.05 %	19217	701.2419

Tabla 2: Observaciones desde julio del año 2010 hasta finales del año 2014

De donde se pueden obtener las siguientes gráficas:

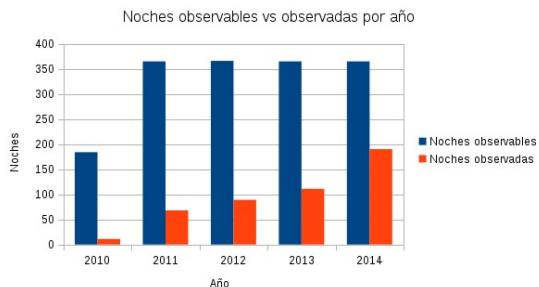


Figura 83: Noches observables vs observadas por año

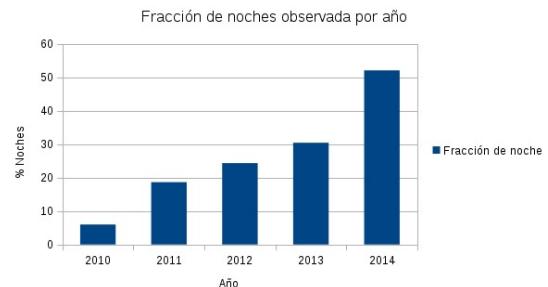


Figura 84: Fracción de noches observadas por año



Figura 85: Imágenes obtenidas por año

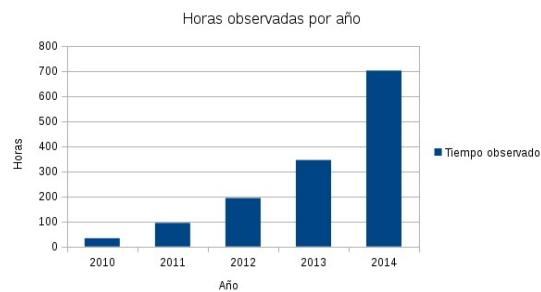


Figura 86: Horas observadas por año

Como se puede observar en las gráficas, el buen funcionamiento del sistema ha fomentado el crecimiento de uso y se han obtenido cada año más resultados. Este hecho ha incentivado a la comunidad científica a hacer más propuestas de observación en este telescopio.

9.1. Patentes

Desde el IEEC-CSIC, se ha solicitado un acta de depósito notarial para OpenROCS. Desde estas instituciones, la protección de software se tramita mediante depósito ante notario en Madrid. Este procedimiento evita el desplazamiento de los autores al Registro de la Propiedad Intelectual, como se hacía anteriormente.

- Acta de depósito notarial relativo al software denominado 'OpenROCS -THE OPEN OBSERVATORY CONTROL SYSTEM', con número de Protocolo 1392/2012 y fecha de depósito 16 de noviembre de 2012.

Este depósito notarial para OpenROCS, aporta un aumento de protección de la propiedad intelectual del proyecto y también facilita los trámites para la realización de transferencia tecnológica entre los institutos de investigación y la industria.

Como nota, indicar que aunque se ha hecho un proceso de patente sobre este software, también se ha liberado como software libre bajo la licencia GPL-3.0. El código fuente esta disponible para ser descargado desde Sourceforge en la siguiente dirección:

- <https://sourceforge.net/projects/openrocs/>

9.2. Resultados científicos

Los resultados científicos obtenidos con el TJO han sido:

1. Spectroscopic confirmation and additional photometry of the M31 nova candidate PNV J00423972+4120117

2014-10-24

G. Sala, P. Rodriguez-Gil, M. Henze, et al.

Astronomer's Telegram, #6616

<http://www.astronomerstelegram.org/?read=6616>

2. Spectroscopy and photometry of the novae M31N 2014-09a and M31N 2014-09b

2014-09-25

E. A. Barsukova, S. Fabrika, A. F. Valeev, et al.
Astronomer's Telegram, #6498
<http://www.astronomerstelegram.org/?read=6498>

3. SN 2013df, a double-peaked I Ib supernova from a compact progenitor and an extended H envelope

2014-09-17

A. Morales-Garoffolo, N. Elias-Rosa, S. Benetti, et al.
Monthly Notices of the Royal Astronomical Society, Volume 445, Issue 2, p.1647-1662
http://adsabs.harvard.edu/cgi-bin/bib_query?arXiv:1409.2784

4. New optical nova candidate in the M 31 disk

2014-07-10

M. Henze, G. Sala, J. Jose, et al.
Astronomer's Telegram, #6305
<http://www.astronomerstelegram.org/?read=6305>

5. Optical and near-infrared observations of SN 2011dh - The first 100 days

2013-05-08

M. Ergon, J. Sollerman, M. Fraser, et al.
Astronomy & Astrophysics, Volume 562, id.A17, 35 pp.
<http://adsabs.harvard.edu/abs/2013arXiv1305.1851E>

6. The 2011 October Draconids outburst. I. Orbital elements, meteoroid fluxes and 21P/Giacobini-Zinner delivered mass to Earth

2013-04-29

J. M. Trigo-Rodríguez, J. M. Madiedo, I. P. Williams, et al.
Monthly Notices of the Royal Astronomical Society, Volume 433, Issue 1, p.560-570
<http://adsabs.harvard.edu/abs/2013arXiv1304.7635T>

9.3. Publicaciones técnicas

Las publicaciones técnicas que se han realizado con el TJO han sido:

1. Observing with the Telescopi Joan Oró

2013, may

Vilardell, F., Colomé, J., Sanz, J., GIL, P., Ribas, I.
Highlights of Spanish Astrophysics VII, pp. 958-958

2. OpenROCS: a software tool to control robotic observatories

2012-09-30

J. Colomé, J. Sanz, F. Vilardell, I. Ribas, P. Gil
Proceedings SPIE, Amsterdam (2012)
<http://adsabs.harvard.edu/abs/2012SPIE.8451E..27C>

We present the Open Robotic Observatory Control System (OpenROCS), an open source software platform developed for the robotic control of telescopes. It acts as a software infrastructure that executes all the necessary processes to implement responses to the system events that appear in the

routine and non-routine operations associated to data-flow and housekeeping control. The OpenROCS software design and implementation provides a high flexibility to be adapted to different observatory configurations and event-action specifications. It is based on an abstract model that is independent of the specific hardware or software and is highly configurable. Interfaces to the system components are defined in a simple manner to achieve this goal. We give a detailed description of the version 2.0 of this software, based on a modular architecture developed in PHP and XML configuration files, and using standard communication protocols to interface with applications for hardware monitoring and control, environment monitoring, scheduling of tasks, image processing and data quality control. We provide two examples of how it is used as the core element of the control system in two robotic observatories: the Joan Oró Telescope at the Montsec Astronomical Observatory (Catalonia, Spain) and the SuperWASP Qatar Telescope at the Roque de los Muchachos Observatory (Canary Islands, Spain).

**SPIE
Astronomical
Instrumentation**

POSTER 8451-80
Session Software and
Cyberinfrastructure for
Astronomy II

OpenROCS: a software tool to control robotic observatories

Pep Colomé^[colome@ieec.cat], Josep Sanz, Francesc Vilardell, Ignasi Ribas, Pere Gil

Institut de Ciències de l'Espai (IEEC-CSIC), Barcelona

Abstract. We present the Open Robotic Observatory Control System (OpenROCS), an open source software platform developed for the robotic control of telescopes. It acts as a software infrastructure that executes all the necessary processes to implement responses to the system events that appear in the routine and non-routine operations associated to data-flow and housekeeping control. The OpenROCS software design and implementation provides a high flexibility to be adapted to different observatory configurations and event-action specifications. It is based on an abstract model that is independent of the specific hardware or software and is highly configurable. Interfaces to the system components are defined in a simple manner to achieve this goal. We give a detailed description of the version 2.0 of this software, based on a modular architecture developed in PHP and XML configuration files, and using standard communication protocols to interface with applications for hardware monitoring and control, environment monitoring, scheduling of tasks, image processing and data quality control. We provide two examples of how it is used as the core element of the control system in two robotic observatories: the Joan Oró Telescope at the Montsec Astronomical Observatory (Catalonia, Spain) and the SuperWASP Qatar Telescope at the Roque de los Muchachos Observatory (Canary Islands, Spain).

Robotics Operation

- Fully autonomous facilities
- Control modes for the execution of science observations and calibration tasks
- Operation modes for the operation of different instruments
- Scheduler for task prioritization → Poster : 8848-59 Session 8
- Nominal Workflow with fully unattended processes
- Task scheduling, Task execution & Data Processing

Control System

- Central application in the telescope control layer
- Software control layer
 - Housekeeping layer: Applications devoted to the environment monitoring and alarm generation, health monitoring and control.
 - End-to-end data flow layer: Software processes for task management, task scheduling, and processing of the acquired data. It also includes software and/or firmware processes for hardware control in the execution of tasks.
- Control system goals
 - Connect the suite of applications involved in the housekeeping and the end-to-end data flow control
 - Administrate the workflow of the observation processes

OpenROCS Generalities

- Modular architecture
- High level of abstraction design motivated by the heterogeneity of the different subsystems
- Highly customizable to use it in robotic observatories
- Operation handled with actions triggered by events

OpenROCS global architecture. One instance is running on each computer at the observatory

OpenROCS Modular Design

- Server module: Handles requests from the other services using TCP/IP
- Broadcast service: Synchronization of multiple instances of OpenROCS in different computers.
- Monitor service: Execution of periodic tasks to perform a continuous evaluation of the system and determine its state.
- Scheduler service: Execution of actions when preconfigured variables change. It is used to control the entire telescope workflow, implementing the data flow events.
- Client module: Provides a direct control of the service status (start/stop/restart/reload).

OpenROCS Implementation

- Programming language: PHP & XML files for configuration.
- Interfaces: Signals and pipe, TCP/IP sockets, UDP/IP sockets, shell commands (SNMP, TALON wrappers).
- Process nodes: Specification of commands, evaluations and flow control used by the Monitor and Scheduler services.
- Configuration: Based on XML standard syntax. XML files define the system configuration, the variables, the data structures, and the event-action pairs that OpenROCS manages.
- Wrapper libraries: TALON, SNMP protocol. INDI available in next version.
- License: GPL-3.0 → <http://sourceforge.net/projects/openrocs>

OpenROCS at TJO and SQT Robotic Telescopes

- TJO: Multipurpose telescope located at the Montsec Astronomical Observatory (OADM, Spain) and endowed with a 0.8-m telescope and two instruments for photometric and spectroscopic data collection. <http://www.oadm.cat>
- SQT: 1-m robotic telescope equipped with a two-arm instrument and housed in a clam-shell dome. It is devoted to exoplanet characterization and follow-up programs with multicolor observations of transients discovered from the real-time reduction of SuperWASP data.
- OpenROCS configuration: Similar configuration in both telescopes. Easy customization thanks to its flexibility and modularity.

TJO and SQT architecture

Conceptual design of XML files for TJO telescope

References

- [1] Baruch, J. E. F., Hedges, D. G., Marshall, J., and Talon, C. J., "Robotic telescopes and the public," *AN*, v.325, 457-461, DOI: 10.1002/and.201100010
- [2] Colomé et al., "Research on schedules for astronomical observatories", Proc. SPIE 8448, these proceedings (2012)
- [3] Clear Sky Institute, "Observatory Control and Astronomical Analysis System. Reference manual for OCAAS version 2.0", software manual (2009)

IEEC ICE

Figura 87: Poster presentado en el SPIE de Amsterdam 2012

3. The Open Robotic Observatory Control System (OpenROCS)

2012

9.4. SuperWASP Qatar Telescope (SQT)

El Telescopio de Seguimiento, SuperWASP Qatar Telescope (SQT), proporcionará fotometría de alta precisión de exoplanetas en tránsito para poder determinar sus propiedades principales, tales como su tamaño y las características orbitales. Trabajará en paralelo con el experimento SuperWASP, que tiene como objetivo buscar exoplanetas en imágenes de gran campo. Este telescopio observará cuidadosamente determinadas estrellas identificadas con SuperWASP para confirmar la naturaleza de sus exoplanetas y estudiar sus propiedades con gran precisión.



Figura 88: Exterior telescopio SuperWASP y SQT



Figura 89: Exterior telescopio SQT

El proyecto SQT, es un proyecto de robotización de un telescopio del fabricante OMI de 1m de diámetro y cuya configuración es muy parecida a la del TJO, pero con algunas diferencias. En este caso, OpenROCS se ha adaptado de forma fácil y rápidamente, dando lugar a un sistema robótico que ya está preparado para funcionar.



Figura 90: Telescopio SQT



Figura 91: Armario de comunicaciones

A continuación se muestran algunas capturas de pantalla del GUI que se implementó para este proyecto y que se comunica en tiempo real con OpenROCS para mostrar el estado del sistema:

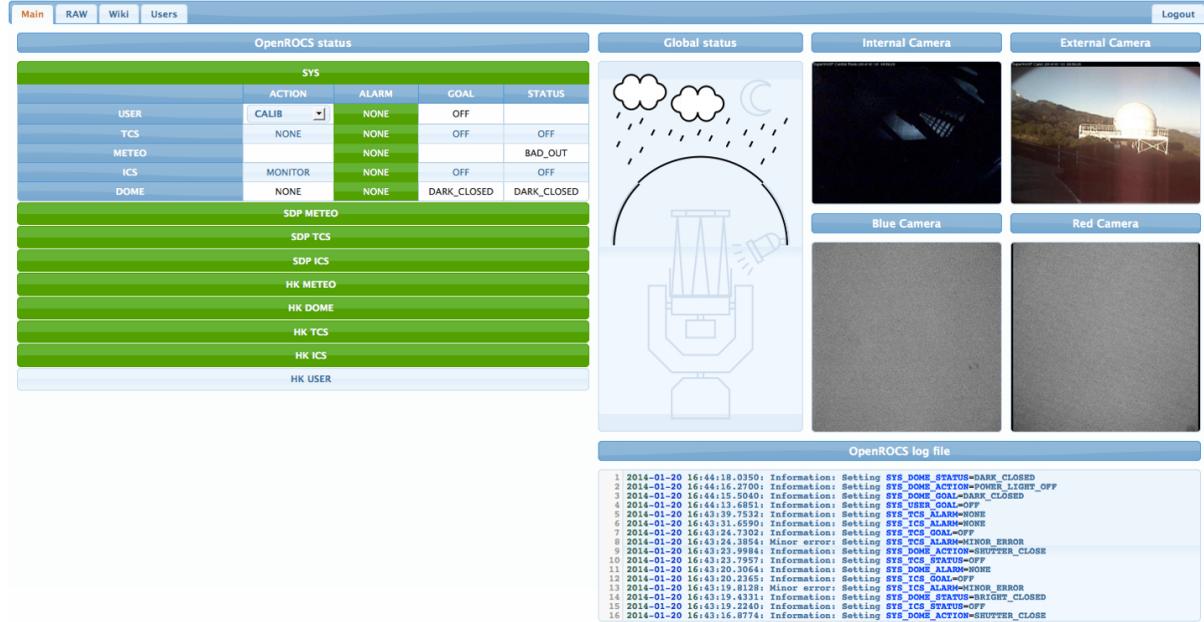


Figura 92: Pantalla de control principal de OpenROCS v2.0 para el telescopio SQT



Figura 93: Pantalla de registros de OpenROCS v2.0 para el telescopio SQT

10. Planificación y costes

En este capítulo, se abordan los aspectos de planificación y valoración económica del proyecto. También se hace un repaso sobre la valoración económica de las instalaciones que albergan al TJO, para dar una idea del orden de magnitud a nivel económico de este proyecto.

10.1. Valoración económica del OAdM y del TJO

La relación de costes detallados para la construcción del proyecto TJO y OAdM ha sido cubierta, en su mayoría, por fondos económicos de la Generalitat de Catalunya (con una significativa contribución de fondos europeos, FEDER). El coste de mantenimiento y gestión van a cargo del IEEC, a través del acuerdo de financiación que este tiene con la Generalitat de Catalunya para llevar a cabo su actividad y, en particular, para la operación de esta infraestructura.



Finalmente, las instituciones que forman parte del IEEC han aportado recursos propios de forma continuada, así como fondos adicionales en momentos puntuales, principalmente para la construcción de nueva instrumentación (p.e., cámara de observación de todo el cielo o espectrógrafo ARES).

A continuación se muestra una tabla con la inversión aproximada en la construcción y compra de equipamiento para el OAdM y el TJO.

Descripción	%
Obra civil (año 2003, fase A)	18 %
Obra civil (año 2007)	22 %
Telescopio OMI 80cm en configuración Ritchey-Chrétien	45 %
Instrumentación (MEIA/ARES/Equipos informáticos)	15 %
Total	100 %

Tabla 3: Inversión en la construcción y equipamiento para el OAdM y el TJO

Notas:

- El total de la inversión fue de **1.328.000,00€**.
- Indicar que la fase B de la obra civil del año 2003, no se ejecutó y posteriormente, en el año 2007, se realizó otro proyecto de adecuación de las instalaciones.
- El coste de la obra civil supuso un **40 %** del total de la inversión mientras que el coste del telescopio + instrumentación supuso un **60 %** del total de la inversión.

La siguiente tabla, muestra el coste de la inversión y el coste de mantenimiento entre los años 2004 y 2013 (ambos incluidos) para el OAdM y el TJO.

Descripción	Precio	%
Inversión	1.328.000,00€	60 %
Mantenimiento del OAdM en el periodo 2004-2013	887.000,00€	40 %
Total	2.215.000,00€	100 %

Tabla 4: Inversión y mantenimiento entre los años 2004 y 2013 para el OAdM y el TJO

10.2. Planificación del proyecto

En la siguiente gráfica se puede observar la distribución de la dedicación respecto las tareas:



Figura 94: Distribución de la dedicación por tareas

La ejecución de este proyecto se planificó con 12 meses de trabajo (200 días contando vacaciones y festividades). El siguiente diagrama de Gantt, muestra la planificación por días:

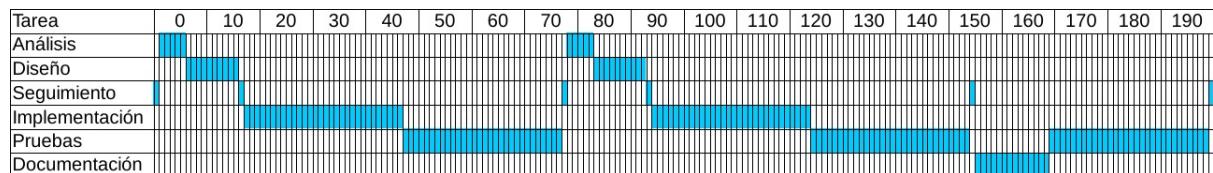


Figura 95: Diagrama de Gantt de la planificación del proyecto

Como se puede ver en este diagrama, la realización de este proyecto se planificó a nivel práctico en 2 fases:

1. En la primera fase: se puso en marcha un sistema que cumpliera los requerimientos mínimos del análisis inicial.
2. En la segunda fase: se hizo otra iteración completa (análisis + diseño + implementación + pruebas) para mejorar el sistema y cumplir con todos los requerimientos marcados por el proyecto.

La decisión de separar el proyecto en 2 fases, fue fruto de la necesidad de tener un prototipo que funcionara y que pudiera ser empleado en el TJO de forma rápida para poder validar su funcionamiento de la forma más fiable y detectar nuevas necesidades. Evidentemente, los números presentados anteriormente a nivel de noches observadas, imágenes obtenidas y tiempo de exposición, son fruto de las 2 iteraciones que se han hecho en este proyecto.

En el siguiente diagrama de Gantt, se puede observar como las tareas de implementación, pruebas y documentación se solaparon para ser más eficientes en el uso del tiempo.

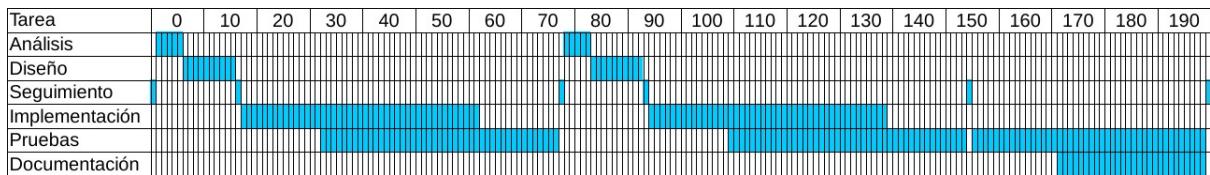


Figura 96: Diagrama de Gantt de la ejecución del proyecto

10.3. Costes de ejecución

Teniendo como base que se destinaron 12 meses (200 días) para realizar este proyecto y fijando un precio hora de 30€/h (este precio incluye los gastos de contratación + vacaciones), se puede concluir con la siguiente tabla:

Tarea	%	Días	Horas	Personas	Precio hora	Total
Análisis	5 %	10	75h	1	30,00€/h	2.250,00€
Diseño	10 %	20	150h	1	30,00€/h	4.500,00€
Seguimiento	3 %	6	45h	1	30,00€/h	1.350,00€
Implementación	30 %	60	450h	1	30,00€/h	13.500,00€
Pruebas	45 %	90	675h	2	30,00€/h	40.500,00€
Documentación	7 %	14	105h	1	30,00€/h	3.150,00€
Total	100 %	200	1500h			65.250,00€

Tabla 5: Costes de ejecución de este proyecto

Como se puede ver en la tabla anterior, todas las tareas han sido llevadas a cabo por una sola persona excepto el paquete de horas destinadas a pruebas, donde ha sido necesaria la interacción directa con uno de los operadores del telescopio.

Este proyecto ha supuesto un coste del:

- **4.91 %** respecto a la inversión del OAdM y TJO.
- **7,36 %** respecto a la partida destinada al mantenimiento del OAdM y TJO.
- **2.95 %** respecto al total de todo el proyecto del OAdM y TJO.

11. Conclusiones y trabajo futuro

11.1. Conclusiones

El nuevo OpenROCS v2.0 ha dado solución al problema de la automatización del control robótico y desatendido del Telescopio Joan Oró. El nuevo software de control ha demostrado ser estable, con capacidad de recuperación ante errores, escalable y adaptable a otros proyectos.

Actualmente ya tenemos OpenROCS instalado y funcionando en dos instalaciones:

- El TJO, caso de estudio y objetivo principal de este proyecto.
- El SuperWASP Qatar Telescope (SQT).

El coste de desarrollo de esta solución informática, de **2.95 %** respecto al total invertido en el OAdM y TJO, y el gran impacto que ha tenido sobre el resultado en las operaciones del Telescopio Joan Oró, justifican el desarrollo de OpenROCS v2.0. Se han conseguido los objetivos marcados por este proyecto y el grado de satisfacción del equipo del OAdM y del TJO es alto, cosa que me anima a continuar y a evolucionar este nuevo sistema de control.

11.2. Trabajo futuro

Con el conocimiento adquirido con esta versión de OpenROCS, se han detectado algunos puntos donde se podrían realizar las siguientes mejoras:

- Mejorar la especificación del lenguaje de configuración (XML) para simplificarlo: actualmente hemos visto, que al no usar atributos en los nodos, los ficheros de configuración son muy grandes y esto hace complicado el mantenimiento del sistema en general. La idea que tenemos es convertir ciertos nodos en atributos para compactar el XML y que de esta manera, tengamos menos líneas y por lo tanto, se simplifique la configuración de todos los ficheros XML.
- Buscar reemplazos para el interprete de PHP que proporcionen mejoras de rendimiento. Actualmente, existen varias propuestas como:
 - HHVM: este proyecto es el resultado de un largo camino de trabajo realizado por el equipo de desarrolladores de Facebook y que permite la ejecución de código PHP con unas mejoras de rendimiento notables. Según algunos tests realizados por el propio equipo de desarrolladores, consiguen ejecutar código entre 2 y 87 veces más rápido que el interprete de Zend.
 - HippyVM: este proyecto es un desarrollo relativamente joven y se basa en el uso de librerías como PyPy y lenguajes como R-Python para conseguir ejecutar código PHP con unas mejoras, según los autores, de como mínimo 2 veces más rápido que HHVM. Este proyecto está todavía en un estado alfa, aunque es interesante ver que hay opciones de reemplazo para el interprete de PHP que proporciona Zend.
- Añadir un algoritmo de Inteligencia artificial para tomar decisiones de forma automática mediante el uso del algoritmo STRIPS: esta mejora permitirá que en lugar de definir todo el workflow del sistema, se definan acciones con requerimientos que deben satisfacerse y que proporcionen un cambio al sistema, de forma que será el propio OpenROCS quien para cada caso, generará las posibles soluciones y elegirá la que menos coste tenga. A continuación se muestra una captura de pantalla de un prototipo que muestra gráficamente la idea en que se basa SRIPS:

Online STRIPS Demonstrator r10

Defined actions

```

1 <root>
  <action>
    <name>power_on</name>
    <require>power-off</require>
    <provide>power-on</provide>
    <weight>1</weight>
  </action>
  <action>
    <name>power_off</name>
    <require>power-on</require>
    <provide>power-off</provide>
    <weight>1</weight>
  </action>
  <action>
    <name>telescope_on</name>
    <require>telescope-off, power-on</require>
    <provide>telescope-on, power-off</provide>
    <weight>1</weight>
  </action>

```

Defined status

```

1 <root>
  <status>
    <name>start</name>
    <vars>power-on, telescope=on, camera=on, dome=open</vars>
  </status>
  <status>
    <name>stop</name>
    <vars>power-off, telescope=off, camera=off, dome=close</vars>
  </status>
  <status>
    <name>science</name>
    <vars>power-on, telescope=on, camera=on, dome=open</vars>
  </status>

```

Full paths graph

Validate the XML file

	From:	To:
Power:	On	Off
Telescope:	On	Off
Dome:	Close	Close
Camera:	On	Off

Validate the XML file

	From:	To:
Status:	Start	Stop

Show original plot

Compute path using actions

Compute path using status

Full paths list

```

camera_off => telescope_off => power_off
dome_open => camera_off => dome_close => telescope_off => power_off

```

Figura 97: Interfaz gráfica para el control de una implementación de STRIPS

12. Anexos

12.1. El fichero config.xml

12.1.1. El nodo <server>

Este nodo define los parámetros del servidor TCP/IP usados por el servicio Servidor y define los stacks que tendrá inicialmente creados el servidor.

```
1  <server>
2    <host>127.0.0.1</host>
3    <port>2323</port>
4    <name>myOpenROCS</name>
5    <stacks>
6      <stack>HK</stack>
7      <stack>SDP</stack>
8      <stack>SYS</stack>
9    </stacks>
10   </server>
```

Código 7: Formato del nodo <server> del fichero config.xml

Notas:

- El sistema, crea por defecto (y es necesario para el control interno de los procesos) sólo un stack (CHLD) para guardar la información relativa a los servicios usados por OpenROCS. Esta estructura se serializa y se guarda en una cadena en formato base64. Esta estructura de datos contiene la siguiente información: PID, HOSTNAME, NODE y NAME.
- El nodo <name> permite establecer un nombre que será útil para el *servicio de broadcast*.

12.1.2. El nodo <broadcast>

Este nodo permite definir los parámetros del servicio broadcast usado por OpenROCS.

```
1  <broadcast>
2    <enabled>true</enabled>
3    <port>2424</port>
4    <discovery>60</discovery>
5    <synchronize>10</synchronize>
6  </broadcast>
```

Código 8: Formato del nodo <broadcast> del fichero config.xml

Notas:

- Se puede desactivar este servicio estableciendo el nodo <enabled> a false.
- El nodo <discovery> establece el intervalo usado por el servicio broadcast para enviar paquetes a la dirección de broadcast.
- El nodo <synchronize> establece el intervalo usado por el servicio de broadcast para replicar los datos entre los diferentes OpenROCS.

12.1.3. El nodo <debug>

Este nodo define los parámetros de depuración de OpenROCS.

```
1  <debug>
2    <comm>false</comm>
3    <signal>false</signal>
4    <process>false</process>
5    <trace>true</trace>
6    <maxlines>1000</maxlines>
7    <percent>50</percent>
8  </debug>
```

Código 9: Formato del nodo <debug> del fichero config.xml

Notas:

- El nodo <maxlines> establece el número máximo de líneas que puede ser guardado en cada fichero de registro (log). Si el fichero contiene más líneas, el sistema hará una rotación del log y se creará un nuevo fichero de log (como lo hacen los sistemas de rotación de logs de los sistemas UNIX). Este parámetro aplica a todos los logs que el sistema usa.
- El nodo <percent> establece el timeout máximo permitido antes de que el sistema haga un reporte de la traza para tareas de depuración en el fichero *debug.log*. Esto es útil para comprobar la calidad de la comunicación del sistema y/o detectar posibles problemas.
- Los otros nodos pueden activar o desactivar prestaciones del registro para cada módulo (recomendado sólo para propósitos de depuración). El sistema de log escribe las salidas hacia la salida estándar (*stdout*).
- El sistema de rotación de los sistemas UNIX, permite renombrar los ficheros de log periódicamente para dejarlo como un fichero antiguo y crear uno nuevo en su lugar. Esta técnica evita que existan ficheros de registro extremadamente grandes.

12.1.4. El nodo <shell>

Este nodo sólo define el comportamiento del histórico para la shell interactiva de *La herramienta Cliente*.

```
1  <shell>
2    <history>$HOME/.openrocs_history</history>
3    <maxlines>1000</maxlines>
4  </shell>
```

Código 10: Formato del nodo <shell> del fichero config.xml

Notas:

- El nodo <maxlines> establece el máximo numero de entradas en el histórico. Si el numero de entradas es mayor que el especificado en este nodo, sólo se guardan las últimas <maxline> líneas.

12.1.5. El nodo <timeout>

Este nodo define los timeouts usados internamente por OpenROCS en las tareas de polling.

```

1 <timeout>
2   <comm>1000000</comm>
3   <childs>1000000</childs>
4   <pipes>3000000</pipes>
5   <wait>5000000</wait>
6   <server>5000000</server>
7   <semaphore>5000000</semaphore>
8 </timeout>

```

Código 11: Formato del nodo <timeout> del fichero config.xml

Nota importante:

- Es recomendable dejar estos valores tal como están, pues estos valores deben guardar coherencia entre sí para garantizar un correcto funcionamiento del sistema.

12.1.6. El nodo <polling>

Este nodo define los tiempos de polling usados internamente por OpenROCS:

```

1 <polling>
2   <comm>1000</comm>
3   <childs>1000</childs>
4   <pipes>1000</pipes>
5   <wait>100000</wait>
6   <server>100000</server>
7   <monitor>100000</monitor>
8   <scheduler>100000</scheduler>
9   <broadcast>100000</broadcast>
10 </polling>

```

Código 12: Formato del nodo <polling> del fichero config.xml

Nota importante:

- Es recomendable dejar estos valores tal como están, pues estos valores deben guardar coherencia entre sí para garantizar un correcto funcionamiento del sistema.

12.1.7. El nodo <retries>

Este nodo define los reintentos usados internamente por OpenROCS cuando una rutina de polling falla.

```

1 <retries>
2   <comm>5</comm>
3   <childs>5</childs>
4   <pipes>5</pipes>
5 </retries>

```

Código 13: Formato del nodo <retries> del fichero config.xml

Nota importante:

- Es recomendable dejar estos valores tal como están, pues estos valores deben guardar coherencia entre sí para garantizar un correcto funcionamiento del sistema.

12.1.8. El nodo <ini_set>

Este nodo es para uso interno sólo. El usuario no tiene que modificar estos valores porque están configurados por el administrador de sistema de acuerdo al perfil del computador que lo ejecuta.

```
1 <ini_set>
2   <session.bug_compat_42>0n</session.bug_compat_42>
3   <register_globals>Off</register_globals>
4   <memory_limit>128M</memory_limit>
5   <max_execution_time>0</max_execution_time>
6   <date.timezone>UTC</date.timezone>
7   <default_charset>UTF-8</default_charset>
8 </ini_set>
```

Código 14: Formato del nodo <ini_set> del fichero config.xml

Nota importante:

- Es recomendable dejar estos valores tal como están, pues estos valores deben guardar coherencia entre si para garantizar un correcto funcionamiento del sistema.

12.1.9. El nodo <putenv>

Permite configurar algunas variables de entorno. Este nodo es importante porque las variables como PATH o LANG usadas por la linea de comandos determinan si se ejecutarán de forma correcta o no y retornaran los mensajes tal como es espera en los monitores y schedulers.

```
1 <putenv>
2   <PATH>/bin:/usr/bin:/usr/local/bin</PATH>
3   <LANG>en_US.UTF-8</LANG>
4 </putenv>
```

Código 15: Formato del nodo <putenv> del fichero config.xml

Nota importante:

- Es recomendable dejar estos valores tal como están, pues estos valores deben guardar coherencia entre si para garantizar un correcto funcionamiento del sistema.

12.2. Los nodos de proceso

12.2.1. El nodo <action>

Este nodo permite ejecutar acciones definidas. Las acciones se pueden definir en varios archivos xml y permiten definir, por ejemplo, funciones, procedimientos y/o macros que se puede llamar desde más de un punto de los nodos de proceso.

La especificación general de las acciones es:

```
1 <actions>
2   <action>
3     <name>myaction1</name>
4     *
5     *   Process nodes
6     *
7   </action>
```

```

8      <action>
9          <name>myaction2</name>
10         *
11         *   Process nodes
12         *
13     </action>
14 </actions>

```

Código 16: Formato del nodo <action> de los nodos de proceso

Para llamar a alguna acción de este archivo, se debe usar la siguiente notación:

```

1  <action>action_file.xml[action_name]</action>

```

Código 17: Ejemplo de como ejecutar a una acción externa desde un nodo <action>

Si se prefiere separar todas las acciones en archivos individuales, se puede crear un único archivo XML que contenga sólo los nodos de proceso. Para ejecutar alguna acción individual, sólo es necesario especificar el nombre del archivo XML sin especificar el action_name:

```

1  <action>action_file.xml</action>

```

Código 18: Ejemplo de como ejecutar a una acción externa desde un nodo <action>

12.2.2. El nodo <shell>

La típica línea de comandos que se puede ejecutar directamente en una consola. Utiliza el shell por defecto del sistema (es recomendable no usar el usuario de root y GNU/Linux utiliza generalmente /bin/bash como shell por defecto para todos los usuarios del sistema).

```

1  <shell>ping -c 3 -i 1 -W 1 $IP_SERVER | grep received</shell>

```

Código 19: Formato del nodo <shell> de los nodos de proceso

Notas:

- Este nodo puede utilizar las variables definidas en el archivo xml/variables.xml.
- El shell por defecto utilizado para ejecutar comandos es el /bin/sh. Se debe saber que la shell que se utiliza es importante para evitar errores con las variables de entorno que se espera tener disponibles.
- Todos los comandos de shell son ejecutados en segundo plano (background) para permitir a OpenROCS controlar la ejecución del proceso y comprobar el tiempo transcurrido utilizado por el proceso y aplicar las reglas si se ha especificado el *nodo timeout*.

El nodo <timeout>

Define el tiempo máximo permitido para la ejecución de la línea de comandos. Puede especificar la cantidad máxima de tiempo deseado para la ejecución de la shell y cuando se consuma el tiempo permitido, que la ejecución se aborte (se envía un SIGTERM y SIGKILL para detener el proceso).

```
1 <timeout>5</timeout>
```

Código 20: Formato del nodo <timeout> de los nodos de proceso

Notas:

- Esta etiqueta no requiere del nodo <ontimeout>. Puede usarse por separado.
- Esta etiqueta lanza una rutina que matará el proceso puesto en marcha y todos los hijos que tenga para evitar procesos zombies, procesos basura o procesos no utilizados.
- Más información sobre los *procesos zombies* en http://en.wikipedia.org/wiki/Zombie_process

El nodo <ontimeout>

En conjunción con el nodo <timeout>, define la acción a hacer cuando se agote el tiempo permitido para la ejecución de un comando.

```
1 <ontimeout>add HK_INTERNET_SERVER_FAIL</ontimeout>
```

Código 21: Formato del nodo <ontimeout> de los nodos de proceso

Notas:

- Este nodo requiere del nodo <timeout>. Si el nodo <timeout> no está definido, se mostrará un error de configuración del fichero que corresponda.

12.2.3. El nodo <php>

Otra opción para ejecutar código PHP directamente y recuperar el stdout y stderr a utilizar. Consulte el siguiente ejemplo para entenderlo:

```
1 <php>microtime(true)</php>
```

Código 22: Formato del nodo <php> de los nodos de proceso

Y un ejemplo completo para una tarea periódica:

```
1 <task>
2   <php>microtime(true)</php>
3   <frequency>1</frequency>
4   <send>update SDP_PHP_MICROTIME=$STDOUT</send>
5 </task>
```

Código 23: Ejemplo del formato de una tarea periódica

Notas:

- Este nodo puede utilizar las variables definidas en el archivo xml/variables.xml.

12.2.4. El nodo <choose>

Define una sentencia de control. Esta característica, permite especificar árboles complejos que pueden poner en práctica todos los requisitos necesarios. La estructura <choose> <when> <otherwise> se inspira en el lenguaje XSLT, que permite especificar usando este concepto simple un árbol de decisiones, como el típico *if then else* utilizado en los lenguajes más comunes de programación. El siguiente ejemplo muestra un ejemplo de uso.

```
1  <choose>
2      <when>
3          <eval>$STDOUT4>0</eval>
4          <send>remove HK INTERNET_SERVER_FAIL</send>
5          <fromiter>3</fromiter> or <fromsec>30</fromsec>
6      </when>
7      <otherwise>
8          <send>add HK INTERNET_SERVER_FAIL</send>
9      </otherwise>
10  </choose>
```

Código 24: Formato del nodo <choose> de los nodos de proceso

El nodo <when>

Este nodo ejecuta su contenido cuando se cumple la evaluación del nodo <eval>

El nodo <eval>

Este nodo retorna cierto o falso al nodo <when> que lo contiene. El contenido de esta etiqueta tienen que ser una expresión de PHP y puede usar todas las variables disponibles:

- \$STDOUT: contiene la cadena de salida de la última ejecución del nodo <shell> (normalmente llamado salida estándar)
- \$STDERR: contiene la cadena de error de la última ejecución del nodo <shell> (normalmente llamado salida de error)
- Variables definidas en el archivo variables.xml
- Las variables disponibles en el servidor (SDP/HK/SYS). Por ejemplo, \$HK_INTERNET_SERVER_FAIL.

Notas:

- Las variables STDOUT y STDERR son definidas como un vector que tendrá como valores el contenido del STDOUT y STDERR separado por espacios.

Por ejemplo:

```
1  <shell>ls -l orocs</shell>
```

Código 25: Ejemplo de uso del nodo <eval>

Debería generar la siguiente salida (si no se ha producido ningún error):

```
1 -rwxrwxr-x 1 sanz sanz 707 sep 11 18:41 orocs
```

Código 26: Ejemplo del resultado del nodo <eval>

En este caso, se puede usar:

```
1 <eval>'$OUTPUT9'=='orocs'</eval>
```

Código 27: Ejemplo del nodo <eval> de los nodos de proceso

La evaluación será cierta y el nodo <when> que lo contiene se ejecutará.

El nodo <fromiter>

Este nodo permite ejecutar tareas con cierto retraso, utilizando como unidad, la iteración de la ejecución. Esto permitiría, por ejemplo, posponer la ejecución de algunos nodos <when> algunas iteraciones. Para aclarar la idea de este nodo, se muestra un ejemplo de uso:

```
1 <fromiter>3</fromiter>
```

Código 28: Formato del nodo <fromiter> de los nodos de proceso

Y el resultado de la ejecución será:

Iteración	1	2	3	4	5	6	7	...
Ejecutado	NO	NO	SI	SI	SI	SI	SI	...

Tabla 6: Resultado del ejemplo de la ejecución del nodo <fromiter>

El nodo <everyiter>

Continuando con el concepto de retraso en las ejecuciones, el ejemplo para este nodo es:

```
1 <everyiter>3</everyiter>
```

Código 29: Formato del nodo <everyiter> de los nodos de proceso

Y el resultado de la ejecución será:

Iteración	1	2	3	4	5	6	7	...
Ejecutado	SI	NO	NO	SI	NO	NO	SI	...

Tabla 7: Resultado del ejemplo de la ejecución del nodo <everyiter>

Notas:

- Imagine que usted necesita ejecutar cada 3 iteraciones algo de código, pero: ¿quiere comenzar en la iteración 2 ?. Para este caso es necesario utilizar los siguientes nodos:

```
1 <fromiter>2</fromiter>
2 <everyiter>3</everyiter>
```

Código 30: Formato del nodo `<fromiter>` usando conjuntamente con `<everyiter>`

Y el resultado de la ejecución será:

Iteración	1	2	3	4	5	6	7	...
Ejecutado	NO	SI	NO	NO	SI	NO	NO	...

Tabla 8: Resultado del ejemplo de la ejecución del nodo `<fromiter>` conjuntamente con `<everyiter>`

El nodo `<untiliter>`

El ejemplo para este nodo es:

```
1 <untiliter>3</untiliter>
```

Código 31: Formato del nodo `<untiliter>` de los nodos de proceso

Y el resultado de la ejecución será:

Iteración	1	2	3	4	5	6	7	...
Ejecutado	SI	SI	SI	NO	NO	NO	NO	...

Tabla 9: Resultado del ejemplo de la ejecución del nodo `<untiliter>`

El nodo `<fromsec>`

Este nodo utiliza el mismo concepto que `<fromiter>`, pero en lugar de utilizar las iteraciones como unidad de contador, aquí se utilizará la marca de tiempo para controlar el tiempo transcurrido desde la primera ejecución y activar correctamente el evento para comenzar la siguiente ejecución. Por ejemplo, con esta nodo, si usted desea comenzar la ejecución cuando un valor es estable durante 60 segundos, el ejemplo a utilizar es:

```
1 <fromsec>60</fromsec>
```

Código 32: Formato del nodo `<fromset>` de los nodos de proceso

El nodo `<everysec>`

Ver el nodo `<fromsec>` y `<everyiter>` para entender este nodo.

El nodo <untilsec>

Ver el nodo <fromsec> y <untiliter> para entender este nodo.

El nodo <otherwise>

El comportamiento del nodo <otherwise> es el mismo que el nodo <when> y sólo se ejecuta si todos los nodos <when> previos se evalúan como falso.

12.2.5. El nodo <send>

Esta etiqueta define un canal de comunicación directa con el servidor de almacenamiento:

- Se puede añadir, modificar y borrar stacks y variables (como se documenta en [La herramienta Cliente](#)).
- Vea la sección [La herramienta Cliente](#) para entender todas las opciones disponibles de este nodo.

Un ejemplo de uso de este nodo es:

```
1 <send>add HK_INTERNET_SERVER_FAIL</send>
2 <send>update SDP_TEMPERATURE_DAVIS_INDOR=$STDOUT1</send>
3 <send>remove HK_INTERNET_SERVER_FAIL</send>
```

Código 33: Formato del nodo <send> de los nodos de proceso

12.2.6. El nodo <log>

El nodo <log> permite al usuario definir en los nodos de proceso los mensajes que se guardarán en el fichero de registro user.log. Este nodo es útil para personalizar el proceso de depuración de la configuración de los monitores y schedulers.

Un ejemplo de uso de este nodo es:

```
1 <log>System running correctly</log>
```

Código 34: Formato del nodo <log> de los nodos de proceso

Otro ejemplo de la salida generada por este comando es este fragmento del fichero user.log:

```
1 2013-11-06 17:35:35.5197: Information: Entering GOAL scheduler
2 2013-11-06 17:35:36.4304: Information: Setting SYS_METEO_GOAL=RUN
3 2013-11-06 17:35:36.5671: Information: Setting SYS_DOME_GOAL=BRIGHT_PARK
4 2013-11-06 17:35:36.6544: Information: Setting SYS_TJO_GOAL=PARK
5 2013-11-06 17:35:36.7449: Information: Setting SYS_MEIA_GOAL=PARK
6 2013-11-06 17:35:36.7881: Information: Leaving GOAL scheduler
7 2013-11-06 17:35:36.9378: Information: Entering GOAL scheduler
8 2013-11-06 17:35:37.8155: Information: Leaving GOAL scheduler
9 2013-11-06 17:35:38.9820: Information: Setting SYS_DOME_ACTION=POWER_LIGHT_ON
10 2013-11-06 17:35:40.6335: Information: Setting SYS_DOME_STATUS=BRIGHT_OFF
11 2013-11-06 17:35:41.1945: Information: Setting SYS_DOME_ACTION=POWER_CIRCUITS_ON
12 2013-11-06 17:35:42.6611: Information: Setting SYS_DOME_STATUS=BRIGHT_CLOSE
13 2013-11-06 17:35:47.4802: Information: Setting SYS_TJO_ACTION=POWER_5V_ON
14 2013-11-06 17:35:48.4570: Information: Setting SYS_TJO_STATUS=STANDBY
15 2013-11-06 17:35:54.0672: Information: Setting SYS_TJO_ACTION=POWER_12V_ON
```

```

16 2013-11-06 17:36:00.4190: Information: Setting SYS_TJO_ACTION=POWER_24V_ON
17 2013-11-06 17:36:07.2619: Information: Setting SYS_TJO_ACTION=TALON_START
18 2013-11-06 17:36:39.9762: Information: Setting SYS_TJO_ACTION=POWER_80V_ON
19 2013-11-06 17:36:41.3993: Information: Setting SYS_TJO_STATUS=ON
20 2013-11-06 17:36:43.4782: Information: Setting SYS_METEO_ACTION=WXD_START
21 2013-11-06 17:36:44.1321: Information: Setting SYS_MEIA_ACTION=CAMERAD_START
22 2013-11-06 17:36:44.9294: Information: Setting SYS_METEO_STATUS=RUN
23 2013-11-06 17:36:47.5647: Information: Setting SYS_TJO_ACTION=HA_HOME
24 2013-11-06 17:37:06.3063: Information: Setting SYS_MEIA_STATUS=PARK
25 2013-11-06 17:37:07.0069: Warning: Setting SYS_MEIA_ALARM=WARNING
26 2013-11-06 17:37:43.0172: Information: Setting SYS_TJO_ACTION=DEC_HOME
27 2013-11-06 17:38:53.5221: Information: Setting SYS_TJO_ACTION=FOCUS_HOME
28 2013-11-06 17:39:25.5415: Information: Setting SYS_TJO_STATUS=HOME
29 2013-11-06 17:39:28.7501: Information: Setting SYS_DOME_ACTION=AZ_HOME
30 2013-11-06 17:39:34.9394: Information: Setting SYS_TJO_ACTION=FOCUS_TRACK
31 2013-11-06 17:39:47.6191: Information: Setting SYS_DOME_STATUS=BRIGHT_HOME
32 2013-11-06 17:39:48.2706: Warning: Setting SYS_TJO_ALARM=WARNING
33 2013-11-06 17:39:52.3821: Information: Setting SYS_DOME_ACTION=AZ_PAUSE
34 2013-11-06 17:39:54.6982: Information: Setting SYS_DOME_STATUS=BRIGHT_PAUSE
35 2013-11-06 17:40:07.4181: Information: Setting SYS_TJO_ALARM=None
36 2013-11-06 17:40:08.0353: Information: Setting SYS_TJO_ACTION=HA_DEC_PAUSE
37 2013-11-06 17:40:14.2934: Information: Setting SYS_TJO_STATUS=INIT
38 2013-11-06 17:40:24.6092: Information: Setting SYS_TJO_ACTION=FOCUS_PAUSE
39 2013-11-06 17:40:26.3131: Information: Setting SYS_TJO_STATUS=PAUSE
40 2013-11-06 17:40:35.9305: Information: Setting SYS_TJO_ACTION=HA_DEC_PARK
41 2013-11-06 17:40:55.0067: Information: Setting SYS_TJO_STATUS=INIT
42 2013-11-06 17:41:05.0833: Information: Setting SYS_TJO_ACTION=FOCUS_TRACK
43 2013-11-06 17:41:06.5811: Information: Setting SYS_TJO_STATUS=PARK
44 2013-11-06 17:41:10.3521: Information: Setting SYS_DOME_ACTION=AZ_PARK
45 2013-11-06 17:41:26.5612: Information: Setting SYS_DOME_STATUS=BRIGHT_PARK
46 2013-11-06 17:45:20.5956: Information: Setting SYS_MEIA_ALARM=None
47 2013-11-06 17:46:31.5848: Information: Entering GOAL scheduler
48 2013-11-06 17:46:32.4079: Information: Setting SYS_DOME_GOAL=BRIGHT_THERMAL
49 2013-11-06 17:46:32.4996: Information: Setting SYS_TJO_GOAL= THERMAL
50 2013-11-06 17:46:32.5880: Information: Leaving GOAL scheduler
51 2013-11-06 17:46:32.7355: Information: Entering GOAL scheduler
52 2013-11-06 17:46:33.5588: Information: Leaving GOAL scheduler
53 2013-11-06 17:46:35.6993: Information: Setting SYS_DOME_ACTION=SHUTTER_OPEN
54 2013-11-06 17:48:42.9872: Information: Setting SYS_DOME_STATUS=BRIGHT_THERMAL
55 2013-11-06 17:48:44.8902: Information: Setting SYS_TJO_ACTION=COVER_OPEN
56 2013-11-06 17:48:47.2691: Information: Setting SYS_DOME_ACTION=AZ_PAUSE
57 2013-11-06 17:49:13.9041: Information: Setting SYS_TJO_STATUS=THERMAL
58 2013-11-06 17:49:23.6285: Information: Setting SYS_TJO_ACTION=HA_DEC_PAUSE
59 2013-11-06 17:49:30.1335: Information: Setting SYS_TJO_STATUS=INIT
60 2013-11-06 17:49:40.4035: Information: Setting SYS_TJO_ACTION=FOCUS_PAUSE
61 2013-11-06 17:49:42.1269: Information: Setting SYS_TJO_STATUS=THERMAL
62 2013-11-06 17:51:04.3482: Information: Setting SYS_DOME_STATUS=STANDBY_THERMAL
63 2013-11-06 17:51:04.8739: Minor error: Setting SYS_DOME_ALARM=MINOR_ERROR
64 2013-11-06 17:51:05.0916: Information: Setting SYS_DOME_ACTION=POWER_LIGHT_ON
65 2013-11-06 17:51:06.2872: Information: Setting SYS_DOME_STATUS=BRIGHT_THERMAL
66 2013-11-06 17:51:07.0326: Information: Setting SYS_DOME_ALARM=None
67 2013-11-06 18:19:04.5326: Information: Entering GOAL scheduler
68 2013-11-06 18:19:05.1028: Information: Leaving GOAL scheduler
69 2013-11-06 18:19:58.7088: Information: Entering GOAL scheduler
70 2013-11-06 18:19:59.2400: Information: Setting SYS_DOME_GOAL=DARK_THERMAL
71 2013-11-06 18:19:59.4573: Information: Leaving GOAL scheduler
72 2013-11-06 18:19:59.6049: Information: Entering GOAL scheduler
73 2013-11-06 18:20:00.0880: Information: Leaving GOAL scheduler
74 2013-11-06 18:20:03.0095: Information: Setting SYS_DOME_ACTION=POWER_LIGHT_OFF
75 2013-11-06 18:20:04.3355: Information: Setting SYS_DOME_STATUS=DARK_THERMAL
76 2013-11-06 21:14:33.9929: Information: Entering GOAL scheduler
77 2013-11-06 21:14:34.4774: Information: Setting SYS_DOME_GOAL=BRIGHT_THERMAL
78 2013-11-06 21:14:34.6099: Information: Leaving GOAL scheduler
79 2013-11-06 21:14:34.7212: Information: Setting SYS_DOME_ACTION=POWER_LIGHT_ON
80 2013-11-06 21:14:34.7998: Information: Entering GOAL scheduler
81 2013-11-06 21:14:35.4449: Information: Leaving GOAL scheduler
82 2013-11-06 21:14:36.0572: Information: Setting SYS_DOME_STATUS=BRIGHT_THERMAL
83 2013-11-06 21:19:39.1809: Information: Entering GOAL scheduler
84 2013-11-06 21:19:39.6162: Information: Setting SYS_DOME_GOAL=DARK_THERMAL
85 2013-11-06 21:19:39.7909: Information: Leaving GOAL scheduler
86 2013-11-06 21:19:39.9389: Information: Entering GOAL scheduler
87 2013-11-06 21:19:40.4178: Information: Leaving GOAL scheduler
88 2013-11-06 21:19:41.9802: Information: Setting SYS_DOME_ACTION=POWER_LIGHT_OFF
89 2013-11-06 21:19:43.2661: Information: Setting SYS_DOME_STATUS=DARK_THERMAL
90 2013-11-06 22:05:09.5340: Information: Entering GOAL scheduler
91 2013-11-06 22:05:10.4537: Information: Setting SYS_DOME_GOAL=BRIGHT_PARK
92 2013-11-06 22:05:10.5452: Information: Setting SYS_TJO_GOAL=PARK
93 2013-11-06 22:05:10.6311: Information: Leaving GOAL scheduler
94 2013-11-06 22:05:10.7791: Information: Entering GOAL scheduler
95 2013-11-06 22:05:11.7729: Information: Leaving GOAL scheduler
96 2013-11-06 22:05:12.6384: Information: Setting SYS_DOME_ACTION=POWER_LIGHT_ON
97 2013-11-06 22:05:13.8896: Information: Setting SYS_DOME_STATUS=BRIGHT_THERMAL
98 2013-11-06 22:05:19.3193: Information: Setting SYS_TJO_ACTION=HA_DEC_PARK

```

```

99 2013-11-06 22:06:30.9555: Information: Setting SYS_TJO_STATUS=INIT
100 2013-11-06 22:06:31.8293: Warning: Setting SYS_TJO_ALARM=WARNING
101 2013-11-06 22:07:11.2175: Minor error: Setting SYS_TJO_ALARM=MINOR_ERROR
102 2013-11-06 22:07:11.6973: Information: Setting SYS_TJO_ACTION=FOCUS_TRACK
103 2013-11-06 22:07:14.7403: Information: Setting SYS_DOME_ACTION=SHUTTER_CLOSE
104 2013-11-06 22:07:24.1123: Information: Setting SYS_TJO_ACTION=HA_DEC_PAUSE
105 2013-11-06 22:07:30.7901: Information: Setting SYS_TJO_ALARM=None
106 2013-11-06 22:07:40.2604: Information: Setting SYS_TJO_ACTION=FOCUS_PAUSE
107 2013-11-06 22:07:41.9538: Information: Setting SYS_TJO_STATUS=THERMAL
108 2013-11-06 22:07:51.3723: Information: Setting SYS_TJO_ACTION=COVER_CLOSE
109 2013-11-06 22:08:19.9777: Information: Setting SYS_TJO_STATUS=PAUSE
110 2013-11-06 22:08:29.5405: Information: Setting SYS_TJO_ACTION=HA_DEC_PARK
111 2013-11-06 22:08:38.6222: Information: Setting SYS_TJO_STATUS=INIT
112 2013-11-06 22:08:48.0312: Information: Setting SYS_TJO_ACTION=FOCUS_TRACK
113 2013-11-06 22:08:49.4274: Information: Setting SYS_TJO_STATUS=PARK
114 2013-11-06 22:09:02.7572: Information: Setting SYS_DOME_STATUS=BRIGHT_PAUSE
115 2013-11-06 22:09:07.1762: Information: Setting SYS_DOME_ACTION=AZ_PARK
116 2013-11-06 22:09:58.8771: Information: Setting SYS_DOME_STATUS=BRIGHT_PARK
117 2013-11-07 01:00:00.6795: Information: Entering GOAL scheduler
118 2013-11-07 01:00:01.2520: Information: Setting SYS_DOME_GOAL=DARK_PARK
119 2013-11-07 01:00:01.4241: Information: Leaving GOAL scheduler
120 2013-11-07 01:00:01.5752: Information: Entering GOAL scheduler
121 2013-11-07 01:00:02.1441: Information: Leaving GOAL scheduler
122 2013-11-07 01:00:04.0483: Information: Setting SYS_DOME_ACTION=POWER_LIGHT_OFF
123 2013-11-07 01:00:05.4611: Information: Setting SYS_DOME_STATUS=DARK_PARK
124 2013-11-07 01:56:48.3986: Information: Entering GOAL scheduler
125 2013-11-07 01:56:48.9254: Information: Leaving GOAL scheduler
126 2013-11-07 01:56:49.0728: Information: Entering GOAL scheduler
127 2013-11-07 01:56:49.5609: Information: Leaving GOAL scheduler
128 2013-11-07 01:57:02.2526: Warning: Setting SYS_TJO_ALARM=WARNING
129 2013-11-07 01:57:13.0916: Information: Setting SYS_TJO_ALARM=None
130 2013-11-07 02:56:24.6792: Information: Entering GOAL scheduler
131 2013-11-07 02:56:25.6383: Information: Setting SYS_DOME_GOAL=BRIGHT_PARK
132 2013-11-07 02:56:25.8702: Information: Leaving GOAL scheduler
133 2013-11-07 02:56:26.0179: Information: Entering GOAL scheduler
134 2013-11-07 02:56:27.0623: Information: Leaving GOAL scheduler
135 2013-11-07 02:56:28.2966: Information: Setting SYS_DOME_ACTION=POWER_LIGHT_ON
136 2013-11-07 02:56:29.8973: Information: Setting SYS_DOME_STATUS=BRIGHT_PARK
137 2013-11-07 02:56:39.1548: Warning: Setting SYS_TJO_ALARM=WARNING
138 2013-11-07 02:56:49.8292: Information: Setting SYS_TJO_ALARM=None
139 2013-11-07 03:38:52.4532: Warning: Setting SYS_TJO_ALARM=WARNING
140 2013-11-07 03:39:03.8193: Information: Setting SYS_TJO_ALARM=None
141 2013-11-07 03:41:08.2848: Information: Entering GOAL scheduler
142 2013-11-07 03:41:08.7644: Information: Setting SYS_DOME_GOAL=DARK_PARK
143 2013-11-07 03:41:08.8960: Information: Leaving GOAL scheduler
144 2013-11-07 03:41:09.0840: Information: Entering GOAL scheduler
145 2013-11-07 03:41:09.6182: Information: Leaving GOAL scheduler
146 2013-11-07 03:41:11.2029: Information: Setting SYS_DOME_ACTION=POWER_LIGHT_OFF
147 2013-11-07 03:41:12.5488: Information: Setting SYS_DOME_STATUS=DARK_PARK
148 2013-11-07 07:12:16.2630: Information: Entering GOAL scheduler
149 2013-11-07 07:12:16.6701: Warning: SYS_USER is unknown. Consistency of GOALS not checked.
150 2013-11-07 07:12:16.9329: Information: Leaving GOAL scheduler
151 2013-11-07 07:12:22.4458: Information: Entering GOAL scheduler
152 2013-11-07 07:12:23.4399: Information: Setting SYS_METEO_GOAL=STOP
153 2013-11-07 07:12:23.5734: Information: Setting SYS_DOME_GOAL=DARK_OFF
154 2013-11-07 07:12:23.6654: Information: Setting SYS_TJO_GOAL=OFF
155 2013-11-07 07:12:23.7530: Information: Setting SYS_MEIA_GOAL=OFF
156 2013-11-07 07:12:23.7969: Information: Leaving GOAL scheduler
157 2013-11-07 07:12:23.9422: Information: Entering GOAL scheduler
158 2013-11-07 07:12:24.8501: Information: Leaving GOAL scheduler
159 2013-11-07 07:12:25.9893: Information: Setting SYS_MEIA_ACTION=CAMERAD_STOP
160 2013-11-07 07:12:28.3566: Information: Setting SYS_MEIA_STATUS=OFF
161 2013-11-07 07:12:30.9345: Information: Setting SYS_METEO_ACTION=WXD_STOP
162 2013-11-07 07:12:32.2274: Information: Setting SYS_METEO_STATUS=STOP
163 2013-11-07 07:12:32.8164: Information: Setting SYS_TJO_ACTION=POWER_80V_OFF
164 2013-11-07 07:12:33.9776: Information: Setting SYS_TJO_STATUS=STANDBY
165 2013-11-07 07:12:43.9713: Information: Setting SYS_TJO_ACTION=TALON_STOP
166 2013-11-07 07:12:52.3752: Information: Setting SYS_TJO_ACTION=POWER_24V_OFF
167 2013-11-07 07:12:58.7561: Information: Setting SYS_TJO_ACTION=POWER_12V_OFF
168 2013-11-07 07:13:05.2291: Information: Setting SYS_TJO_ACTION=POWER_5V_OFF
169 2013-11-07 07:13:06.2920: Information: Setting SYS_TJO_STATUS=OFF
170 2013-11-07 07:13:08.6453: Information: Setting SYS_DOME_STATUS=DARK_CLOSE
171 2013-11-07 07:13:10.4202: Information: Setting SYS_DOME_ACTION=POWER_CIRCUITS_OFF
172 2013-11-07 07:13:12.1761: Information: Setting SYS_DOME_STATUS=DARK_OFF

```

Código 35: Fragmento del fichero user.log generado por el TJO en una noche de observación

12.3. Sistema de control del telescopio

12.3.1. talon_fifo

```
1 #!/usr/bin/php -q
2 <?php
3 // FUNCTIONS COPIED FROM SALTOS
4 function capture_next_error() {
5     global $_ERROR_HANDLER;
6     if(!isset($_ERROR_HANDLER["level"])) show_php_error(array("phpperor"=>"error_handler ←
7         without levels availables"));
8     $_ERROR_HANDLER["level"]++;
9     array_push($_ERROR_HANDLER["msg"], "");
10 }
11 function get_clear_error() {
12     global $_ERROR_HANDLER;
13     if($_ERROR_HANDLER["level"]<=0) show_php_error(array("phpperor"=>"error_handler without ←
14         levels availables"));
15     $_ERROR_HANDLER["level"]--;
16     return array_pop($_ERROR_HANDLER["msg"]);
17 }
18 function do_message_error($array,$format) {
19     static $dict=array(
20         "html"=>array(array("<h1>","</h1>"),array("<p>","</p>"),"<br/>"),
21         "text"=>array(array("*****"," *****\n"),array("", "\n"),"\n")
22     );
23     if(!isset($dict[$format])) die("Unknown format $format");
24     $msg=$array();
25     if(isset($array["phpperor"])) $msg []=array("PHP Error",$array["phpperor"]);
26     if(isset($array["xmlerror"])) $msg []=array("XML Error",$array["xmlerror"]);
27     if(isset($array["dberror"])) $msg []=array("DB Error",$array["dberror"]);
28     if(isset($array["emailerror"])) $msg []=array("EMAIL Error",$array["emailerror"]);
29     if(isset($array["fileerror"])) $msg []=array("FILE Error",$array["fileerror"]);
30     if(isset($array["source"])) $msg []=array("XML Source",$array["source"]);
31     if(isset($array["exception"])) $msg []=array("Exception",$array["exception"]);
32     if(isset($array["details"])) $msg []=array("Details",$array["details"]);
33     if(isset($array["query"])) $msg []=array("Query",$array["query"]);
34     if(isset($array["backtrace"])) {
35         $backtrace=$array["backtrace"];
36         array_walk($backtrace,"__debug_backtrace_helper");
37         $msg []=array("Backtrace",implode($dict[$format][2],$backtrace));
38     }
39     array_walk($msg,"__do_message_error_helper",$dict[$format]);
40     $msg=implode($msg);
41     return $msg;
42 }
43
44 function __debug_backtrace_helper(&$item,$key) {
45     $item="${$key} => ".$item["function"].(isset($item["class"])? " (in class ".$item["class"].←
46         ")":").((isset($item["file"])) && isset($item["line"]))?" (in file ".$item["file"]."←
47         at line ".$item["line"]." )":");
48 }
48 function __do_message_error_helper(&$item,$key,$dict) {
49     $item=$dict[0][0].$item[0].$dict[0][1].$dict[1][0].$item[1].$dict[1][1];
50 }
51
52 function show_php_error($array=null) {
53     global $_ERROR_HANDLER;
54     static $backup=null;
55     if(is_null($array)) $array=$backup;
56     if(is_null($array)) return;
57     // REFUSE THE DEPRECATED WARNINGS
58     if(isset($array["phpperor"])) {
59         $pos1=stripos($array["phpperor"],"function");
60         $pos2=stripos($array["phpperor"],"deprecated");
61         if($pos1!==false && $pos2!==false) return;
62     }
63     // ADD BACKTRACE IF NOT FOUND
64     if(!isset($array["backtrace"])) $array["backtrace"]=debug_backtrace();
65     // CREATE THE MESSAGE ERROR USING PLAIN TEXT
66     $msg=do_message_error($array,"text");
67     // CHECK IF CAPTURE ERROR WAS ACTIVE
68     if($_ERROR_HANDLER["level"]>0) {
69         $old=array_pop($_ERROR_HANDLER["msg"]);
70         array_push($_ERROR_HANDLER["msg"],$old.$msg);
71         $backup=$array;
72         return;
```

```

73 }
74 // ADD THE MESSAGE TO THE ERROR LOG FILE
75 //~ addlog($msg,__ERROR_LOG__);
76 // DUMP TO STDOUT
77 while(ob_get_level()) ob_end_clean(); // TRICK TO CLEAR SCREEN
78 echo $msg;
79 die();
80 }

81 function __error_handler($type,$message,$file,$line) {
82     $backtrace=debug_backtrace();
83     array_shift($backtrace);
84     show_php_error(array("phperror"=>"${message} (code ${type})","details"=>"Error on file '$file' at line ${line}","backtrace"=>$backtrace));
85 }
86 }

87 function __exception_handler($e) {
88     $backtrace=$e->getTrace();
89     show_php_error(array("exception"=>$e->getMessage()." (code ".$e->getCode()." )","details"=>
90                         ="Error on file '". $e->getFile()." at line ".$e->getLine(),"backtrace"=>$backtrace));
91 }
92 }

93 function __shutdown_handler() {
94     semaphore_shutdown();
95     $error=error_get_last();
96     $types=array(E_ERROR,E_PARSE,E_CORE_ERROR,E_COMPILE_ERROR,E_USER_ERROR,<-
97                 E_RECOVERABLE_ERROR);
98     if(is_array($error) && isset($error["type"])) && in_array($error["type"],$types)) {
99         global $_ERROR_HANDLER;
100        $_ERROR_HANDLER=array("level"=>0,"msg"=>array());
101        $backtrace=debug_backtrace();
102        show_php_error(array("phperror"=>"${error["message"]}", "details"=>"Error on file '$error["file"]' at line ${error["line"]}", "backtrace"=>$backtrace));
103    }
104 }

105 function program_error_handler() {
106     global $_ERROR_HANDLER;
107     $_ERROR_HANDLER=array("level"=>0,"msg"=>array());
108     error_reporting(0);
109     set_error_handler("__error_handler");
110     set_exception_handler("__exception_handler");
111     register_shutdown_function('__shutdown_handler');
112 }

113 function init_random() {
114     static $init=false;
115     if($init) return;
116     srand((float)microtime(true)*1000000);
117     $init=true;
118 }

119 function current_datetime($offset=0) {
120     return current_date($offset)." ".current_time($offset);
121 }

122 function current_date($offset=0) {
123     return date("Y-m-d",time()+$offset);
124 }

125 function current_time($offset=0) {
126     return date("H:i:s",time()+$offset);
127 }

128 function current_datetime_decimals($offset=0) {
129     return current_datetime($offset).".current_decimals($offset)";
130 }

131 function current_decimals($offset=0) {
132     $decimals=explode(".","microtime(true)+$offset");
133     return substr((isset($decimals[1])?$decimals[1]:"")."0000",0,4);
134 }

135 function semaphore_acquire($file) {
136     return __semaphore_helper(__FUNCTION__,$file);
137 }

138 function semaphore_release($file) {
139     return __semaphore_helper(__FUNCTION__,$file);
140 }

141 function semaphore_shutdown() {
142 }
```

```

151     return __semaphore_helper(__FUNCTION__,null);
152 }
153
154 function __semaphore_helper($fn,$file) {
155     static $stack=array();
156     if(strpos($fn,"acquire")!==false) {
157         $hash=md5($file);
158         if(!isset($stack[$hash])) $stack[$hash]=null;
159         if($stack[$hash]) return false;
160         init_random();
161         while(1) {
162             capture_next_error();
163             $stack[$hash]=fopen($file,"a");
164             get_clear_error();
165             if($stack[$hash]) break;
166             usleep_protected(rand(0,1000));
167         }
168         chmod_protected($file,0666);
169         touch_protected($file);
170         while(1) {
171             capture_next_error();
172             $result=flock($stack[$hash],LOCK_EX|LOCK_NB);
173             get_clear_error();
174             if($result) break;
175             usleep_protected(rand(0,1000));
176         }
177         ftruncate($stack[$hash],0);
178         fwrite($stack[$hash],getmypid());
179         return true;
180     } elseif(strpos($fn,"release")!==false) {
181         $hash=md5($file);
182         if(!isset($stack[$hash])) $stack[$hash]=null;
183         if(!$stack[$hash]) return false;
184         capture_next_error();
185         flock($stack[$hash],LOCK_UN);
186         get_clear_error();
187         capture_next_error();
188         fclose($stack[$hash]);
189         get_clear_error();
190         $stack[$hash]=null;
191         return true;
192     } elseif(strpos($fn,"shutdown")!==false) {
193         foreach($stack as $hash=>$val) {
194             if($stack[$hash]) {
195                 capture_next_error();
196                 flock($stack[$hash],LOCK_UN);
197                 get_clear_error();
198                 capture_next_error();
199                 fclose($stack[$hash]);
200                 get_clear_error();
201                 $stack[$hash]=null;
202             }
203         }
204         return true;
205     }
206     return false;
207 }
208
209 function usleep_protected($usec) {
210     $socket=socket_create(AF_UNIX,SOCK_STREAM,0);
211     $read=null;
212     $write=null;
213     $except=array($socket);
214     capture_next_error();
215     $time1=microtime(true);
216     socket_select($read,$write,$except,intval($usec/1000000),intval($usec%1000000));
217     $time2=microtime(true);
218     get_clear_error();
219     return ($time2-$time1)*1000000;
220 }
221
222 function chmod_protected($file,$mode) {
223     capture_next_error();
224     ob_start();
225     chmod($file,$mode);
226     $error1=ob_get_clean();
227     $error2=get_clear_error();
228     return $error1.$error2;
229 }
230
231 function touch_protected($file) {
232     capture_next_error();
233     ob_start();

```

```

234     touch($file);
235     $error1=ob_get_clean();
236     $error2=get_clear_error();
237     return $error1.$error2;
238 }
239
240 // LOAD ALL ALIAS
241 $alias=array();
242 $file1=getenv("HOME")."./talon_alias";
243 if(file_exists($file1)) $alias=array_merge($alias,file($file1));
244 $file2=dirname($_SERVER["SCRIPT_NAME"])."./talon_alias";
245 if(file_exists($file2)) $alias=array_merge($alias,file($file2));
246 foreach($alias as $key=>$val) {
247     unset($alias[$key]);
248     if(substr($val,0,1)=="#") $val="";
249     if(strpos($val,"=")==false) $val="";
250     if($val!="") $alias[strtok($val,"=")]=trim(strtok(""));
251 }
252
253 // CHECK FIRST ARGUMENT
254 if(isset($argv[1]) && isset($alias[$argv[1]])) {
255     $temp=explode(' ', $argv[0]." ".$alias[$argv[1]]);
256     $param=array_splice($temp,2);
257     foreach($param as $key=>$val) {
258         unset($param[$key]);
259         $param["\$".($key+1)]=$val;
260     }
261     $param=array_reverse($param, true);
262     $argv=array();
263     foreach($temp as $key=>$val) {
264         if($key%2==0) {
265             $temp2=explode(" ",$val);
266             foreach($temp2 as $val) {
267                 if($val!="") $argv[]=str_replace(array_keys($param),array_values($param),$val);
268             }
269         } else {
270             if($val!="") $argv[]=str_replace(array_keys($param),array_values($param),$val);
271         }
272     }
273     $argc=count($argv);
274 }
275
276 // ACQUIRE SEMAPHORE
277 program_error_handler();
278 $temp=array();
279 if(isset($argv[1])) $temp[]=$argv[1];
280 if(isset($argv[3])) $temp[]=$argv[3];
281 if(count($temp)>0) $semaphore_file="/tmp/.md5(serialized($temp)).sem";
282 if(isset($semaphore_file)) semaphore_acquire($semaphore_file);
283
284 // FLUSH ACTION
285 if($argc>=4) {
286     if(!file_exists($argv[3])) die("Fifo to read not found\n");
287     capture_next_error();
288     $fp=fopen($argv[3],"r+");
289     get_clear_error();
290     if($fp==false) die("Could not open the fifo to read\n");
291     stream_set_blocking($fp, false);
292     $buffer="";
293     while($char=fread($fp,1024)) $buffer.=$char;
294     fclose($fp);
295 }
296
297 // WRITE ACTION
298 if($argc>=3) {
299     if(!file_exists($argv[1])) die("Fifo to write not found\n");
300     capture_next_error();
301     $fp=fopen($argv[1],"w+");
302     get_clear_error();
303     if($fp==false) die("Could not open the fifo to write\n");
304     stream_set_blocking($fp, false);
305     $argv[2]=trim($argv[2]);
306     $argv[2]=str_replace("\n","\n",$argv[2]);
307     $argv[2]=explode("\n",$argv[2]);
308     foreach($argv[2] as $temp) {
309         $temp2=strtok($temp," ");
310         switch($temp2) {
311             case "sleep":
312                 sleep(strtok(""));
313                 break;
314             default:
315                 fwrite($fp,$temp."\n");

```

```

316             break;
317         }
318     }
319     fclose($fp);
320 }
321
322 // READ ACTION
323 if($argc>=4) {
324     if(!file_exists($argv[3])) die("Fifo to read not found\n");
325     capture_next_error();
326     $fp=fopen($argv[3],"r+");
327     get_clear_error();
328     if($fp==false) die("Could not open the fifo to read\n");
329     stream_set_blocking($fp,false);
330     $index=4;
331     if($index<$argc && is_numeric($argv[$index])) {
332         $timeout=microtime(true)+$argv[$index];
333         $timeout2=$argv[$index];
334         $index++;
335     }
336     $found=0;
337     while(!$found) {
338         $buffer="";
339         while($char=fread($fp,1024)) $buffer.=$char;
340         $buffer=trim($buffer);
341         if($buffer!="") echo "$buffer\n";
342         if($index<$argc) {
343             for($i=$index;$i<$argc;$i++) {
344                 if(strpos($buffer,$argv[$i])!==false) {
345                     $found=1;
346                     break;
347                 }
348             }
349         } else {
350             if($buffer!="") $found=1;
351         }
352         if(!$found) {
353             usleep_protected(1000);
354             if(isset($timeout)) {
355                 if(microtime(true)>=$timeout) {
356                     echo "Timeout '$timeout2' reached\n";
357                     break;
358                 }
359             }
360         }
361     }
362     fclose($fp);
363 }
364
365 // RELEASE SEMAPHORE
366 if(isset($semaphore_file)) semaphore_release($semaphore_file);
367
368 // HELP
369 if($argc<3) {
370     echo "Usage: ".basename($argv[0])."\n";
371     echo "\talias [arguments]\n";
372     echo "\tfifo_to_write command_to_write\n";
373     echo "\t[fifo_to_read] [timeout_in_seconds] [command_to_read] [command_to_write] ...\n";
374     echo "\n";
375     echo "Available alias:\n";
376     foreach($alias as $key2=>$val2) {
377         echo "\t$key2\n";
378         $temp=explode(' ', $val2);
379         $pos=1;
380         foreach($temp as $key=>$val) {
381             if($key%2==0) {
382                 $temp2=explode(" ", $val);
383                 foreach($temp2 as $val) {
384                     if($val!="") {
385                         echo "\t\t[$pos] $val\n";
386                         $pos++;
387                     }
388                 }
389             } else {
390                 if($val!="") {
391                     echo "\t\t[$pos] $val\n";
392                     $pos++;
393                 }
394             }
395         }
396     }
397     echo "\n";
398     die();
}

```

```

399 }
400 ?>

```

Código 36: Contenido del script talon_fifo

12.3.2. talon_alias

```

1 # CCD_VIS commands
2 camsky=/usr/local/telescope/comm/Camera.in "Expose 0+0x2048x2048 1x1 $1 1 0 $2\nOBJECT\←
   nCOMMENT\nCOMMENT\nOBSERVER" /usr/local/telescope/comm/Camera.out $1+5 error unknown ←
   created "timed out"
3 camsubsky=/usr/local/telescope/comm/Camera.in "Expose $1+$2x$3x$4 1x1 $5 1 0 $6\nSOURCE\←
   nCOMMENT\nTITLE\nOBSERVER" /usr/local/telescope/comm/Camera.out $1+5 error unknown ←
   created "timed out"
4 cambias=/usr/local/telescope/comm/Camera.in "Expose 0+0x2048x2048 1x1 0 0 0 $1\nSOURCE\←
   nCOMMENT\nTITLE\nOBSERVER" /usr/local/telescope/comm/Camera.out 5 error unknown created ←
   "timed out"
5 camdark=/usr/local/telescope/comm/Camera.in "Expose 0+0x2048x2048 1x1 $1 0 0 $2\nSOURCE\←
   nCOMMENT\nTITLE\nOBSERVER" /usr/local/telescope/comm/Camera.out $1+5 error unknown ←
   created "timed out"
6 camstatus=/usr/local/telescope/comm/Camera.in Temperature /usr/local/telescope/comm/Camera.←
   out 5 error temperature
7
8 # MEIA commands
9 meiasky=/usr/local/telescope/comm/Camera.in "Expose 0+0x2048x2048 1x1 $1 1 0 $2\nSOURCE\←
   nCOMMENT\nTITLE\nOBSERVER" /usr/local/telescope/comm/Camera.out $1+5 error unknown ←
   created "timed out"
10 meiasubsky=/usr/local/telescope/comm/Camera.in "Expose $1+$2x$3x$4 1x1 $5 1 0 $6\nSOURCE\←
   nCOMMENT\nTITLE\nOBSERVER" /usr/local/telescope/comm/Camera.out $1+5 error unknown ←
   created "timed out"
11 meiabias=/usr/local/telescope/comm/Camera.in "Expose 0+0x2048x2048 1x1 0 0 0 $1\nSOURCE\←
   nCOMMENT\nTITLE\nOBSERVER" /usr/local/telescope/comm/Camera.out 5 error unknown created ←
   "timed out"
12 meiadark=/usr/local/telescope/comm/Camera.in "Expose 0+0x2048x2048 1x1 $1 0 0 $2\nSOURCE\←
   nCOMMENT\nTITLE\nOBSERVER" /usr/local/telescope/comm/Camera.out $1+5 error unknown ←
   created "timed out"
13 meiastatus=/usr/local/telescope/comm/Camera.in Temperature /usr/local/telescope/comm/Camera.←
   out 5 error temperature
14
15 # Camera commands (mainly for testing)
16 camerasky=/usr/local/telescope/comm/Camera.in "Expose 0+0x512x768 1x1 $1 1 0 $2\nSOURCE\←
   nCOMMENT\nTITLE\nOBSERVER" /usr/local/telescope/comm/Camera.out $1+5 error unknown ←
   created "timed out"
17 camerabias=/usr/local/telescope/comm/Camera.in "Expose 0+0x512x768 1x1 0 0 0 $1\nSOURCE\←
   nCOMMENT\nTITLE\nOBSERVER" /usr/local/telescope/comm/Camera.out 5 error unknown created ←
   "timed out"
18 cameradark=/usr/local/telescope/comm/Camera.in "Expose 0+0x512x768 1x1 $1 0 0 $2\nSOURCE\←
   nCOMMENT\nTITLE\nOBSERVER" /usr/local/telescope/comm/Camera.out $1+5 error unknown ←
   created "timed out"
19 cameratest=/usr/local/telescope/comm/Camera.in "Expose 0+0x512x512 1x1 1 1 0 $1\nSOURCE\←
   nCOMMENT\nTITLE\nOBSERVER" /usr/local/telescope/comm/Camera.out 60 error unknown created←
   "timed out"
20 cameranewton=/usr/local/telescope/comm/Camera.in "Expose 0+0x2048x512 1x1 1 1 0 $1\nSOURCE\←
   nCOMMENT\nTITLE\nOBSERVER" /usr/local/telescope/comm/Camera.out 60 error unknown created←
   "timed out"
21 cameraikoni=/usr/local/telescope/comm/Camera.in "Expose 0+0x2088x2048 1x1 1 1 0 $1\nSOURCE\←
   nCOMMENT\nTITLE\nOBSERVER" /usr/local/telescope/comm/Camera.out 60 error unknown created←
   "timed out"
22 cameraexpose=/usr/local/telescope/comm/Camera.in "Expose $1+$2x$3x$4 $5x$6 $7 $8 $9 $10\n$11\←
   n$12\n$13\n$14" /usr/local/telescope/comm/Camera.out 60 error unknown created "timed out"←
   "
23 camerastop=/usr/local/telescope/comm/Camera.in Stop /usr/local/telescope/comm/Camera.out 60 "←
   stop complete"
24 camerareset=/usr/local/telescope/comm/Camera.in Reset /usr/local/telescope/comm/Camera.out 60←
   "reset complete"
25 cameratemperature=/usr/local/telescope/comm/Camera.in Temperature /usr/local/telescope/comm/←
   Camera.out 1 "current temperature"
26
27 # Telescope commands
28 telhome=/usr/local/telescope/comm/Tel.in homeHD /usr/local/telescope/comm/Tel.out 90 "scope ←
   homing complete"
29 telhomeh=/usr/local/telescope/comm/Tel.in homeH /usr/local/telescope/comm/Tel.out 75 "scope ←
   homing complete"
30 telhomed=/usr/local/telescope/comm/Tel.in homeD /usr/local/telescope/comm/Tel.out 75 "scope ←
   homing complete"
31 telstow=/usr/local/telescope/comm/Tel.in Stow /usr/local/telescope/comm/Tel.out 60 "slew ←
   complete"

```

```

52 telstop=/usr/local/telescope/comm/Tel.in Stop /usr/local/telescope/comm/Tel.out 3 "stop ↵
      complete"
53 telstatus=/usr/local/telescope/comm/Tel.in status /usr/local/telescope/comm/Tel.out 3 "←
      position" "Elevation"
54 #tellimits=/usr/local/telescope/comm/Tel.in Limits /usr/local/telescope/comm/Tel.out
55
56 # Telescope movements
57 telmove=/usr/local/telescope/comm/Tel.in "j$1" /usr/local/telescope/comm/Tel.out 10 "paddle ←
      command"
58 telmovenorth=/usr/local/telescope/comm/Tel.in "jN" /usr/local/telescope/comm/Tel.out 10 "←
      paddle command up"
59 telmovesouth=/usr/local/telescope/comm/Tel.in "jS" /usr/local/telescope/comm/Tel.out 10 "←
      paddle command down"
60 telmovewest=/usr/local/telescope/comm/Tel.in "jW" /usr/local/telescope/comm/Tel.out 10 "←
      paddle command cw"
61 telmoveeast=/usr/local/telescope/comm/Tel.in "jE" /usr/local/telescope/comm/Tel.out 10 "←
      paddle command ccw"
62 telmovestop=/usr/local/telescope/comm/Tel.in "j0" /usr/local/telescope/comm/Tel.out 10 "←
      paddle command stop"
63 telstepnorth=/usr/local/telescope/comm/Tel.in "jN\nsleep 1\nj0" /usr/local/telescope/comm/Tel←
      .out 10 "paddle command stop"
64 telstepsouth=/usr/local/telescope/comm/Tel.in "jS\nsleep 1\nj0" /usr/local/telescope/comm/Tel←
      .out 10 "paddle command stop"
65 telstepwest=/usr/local/telescope/comm/Tel.in "jW\nsleep 1\nj0" /usr/local/telescope/comm/Tel←
      .out 10 "paddle command stop"
66 telstepeast=/usr/local/telescope/comm/Tel.in "jE\nsleep 1\nj0" /usr/local/telescope/comm/Tel←
      .out 10 "paddle command stop"
67 telsidereal2=/usr/local/telescope/comm/Tel.in "jW 100 \nsleep 20\nj0" /usr/local/telescope/←
      comm/Tel.out 3 "paddle command stop"
68
69 # Telescope tracking
70 teltrackepoch=/usr/local/telescope/comm/Tel.in "RA:$1 Dec:$2 Epoch:$3" /usr/local/telescope/←
      comm/Tel.out 90 "now tracking" "error"
71 teltrackeo2=/usr/local/telescope/comm/Tel.in "RA:$1 Dec:$2" /usr/local/telescope/comm/Tel.out←
      90 "now tracking" "error"
72
73 # Telescope slewing
74 telslewaltaz=/usr/local/telescope/comm/Tel.in "Alt:$1 Az:$2" /usr/local/telescope/comm/Tel.←
      out 90 "slew complete"
75 telslewhadec=/usr/local/telescope/comm/Tel.in "HA:$1 Dec:$2" /usr/local/telescope/comm/Tel.←
      out 90 "slew complete"
76
77 # Telescope relative movement
78 telxdelta=/usr/local/telescope/comm/Tel.in "xdelta($1,$2)" /usr/local/telescope/comm/Tel.out ←
      30 "encoder steps" "now tracking" "lost tracking"
79
80 # Focus commands
81 focushome=/usr/local/telescope/comm/Focus.in home /usr/local/telescope/comm/Focus.out 60 "←
      focus offset complete"
82 focuscheck=/usr/local/telescope/comm/Focus.in " " /usr/local/telescope/comm/Focus.out 5 "←
      autofocus offset complete" "focus offset complete"
83 focusauto=/usr/local/telescope/comm/Focus.in auto /usr/local/telescope/comm/Focus.out 10 "←
      autofocus offset complete" "Auto-focus enabled"
84 focusstepin=/usr/local/telescope/comm/Focus.in "j+\nsleep 1\nj0" /usr/local/telescope/comm/←
      Focus.out 6 "stop complete"
85 focusstepout=/usr/local/telescope/comm/Focus.in "j-\nsleep 1\nj0" /usr/local/telescope/comm/←
      Focus.out 6 "stop complete"
86 focusin=/usr/local/telescope/comm/Focus.in j+ /usr/local/telescope/comm/Focus.out 60 "paddle ←
      command in"
87 focusout=/usr/local/telescope/comm/Focus.in j- /usr/local/telescope/comm/Focus.out 60 "paddle←
      command out"
88 focusstop=/usr/local/telescope/comm/Focus.in j0 /usr/local/telescope/comm/Focus.out 3 "stop ←
      complete"
89 focusoffset=/usr/local/telescope/comm/Focus.in $1 /usr/local/telescope/comm/Focus.out 60 "←
      focus offset complete"
90 focusstatus=/usr/local/telescope/comm/Focus.in status /usr/local/telescope/comm/Focus.out 3
91
92 # Filter commands
93 filterhome=/usr/local/telescope/comm/Filter.in home /usr/local/telescope/comm/Filter.out 75 "←
      filter in place"
94 filterright=/usr/local/telescope/comm/Filter.in j+ /usr/local/telescope/comm/Filter.out 60 "←
      filter in place"
95 filterleft=/usr/local/telescope/comm/Filter.in j- /usr/local/telescope/comm/Filter.out 60 "←
      filter in place"
96 filterset=/usr/local/telescope/comm/Filter.in $1 /usr/local/telescope/comm/Filter.out 60 "←
      filter in place"
97 filterstatus=/usr/local/telescope/comm/Filter.in status /usr/local/telescope/comm/Filter.out ←
      3
98
99 # Dome commands
100 domehome=/usr/local/telescope/comm/Dome.in home /usr/local/telescope/comm/Dome.out 150 "home ←
      complete"

```

```

81 domeopen=/usr/local/telescope/comm/Dome.in open /usr/local/telescope/comm/Dome.out 150 "open ↵
     complete"
82 domeclose=/usr/local/telescope/comm/Dome.in close /usr/local/telescope/comm/Dome.out 150 "↵
     close complete"
83 domeslew=/usr/local/telescope/comm/Dome.in Az:$1 /usr/local/telescope/comm/Dome.out 150 "↵
     azimuth command complete" "error"
84 domestopleft=/usr/local/telescope/comm/Dome.in "j-\nsleep 3\nj0" /usr/local/telescope/comm/←
     Dome.out 6 "paddle command stop"
85 domestopright=/usr/local/telescope/comm/Dome.in "j+\nsleep 3\nj0" /usr/local/telescope/comm/←
     Dome.out 6 "paddle command stop"
86 domeright=/usr/local/telescope/comm/Dome.in j+ /usr/local/telescope/comm/Dome.out 150 "paddle←
     command cw"
87 domeleft=/usr/local/telescope/comm/Dome.in j- /usr/local/telescope/comm/Dome.out 150 "paddle ←
     command ccw"
88 domestop=/usr/local/telescope/comm/Dome.in stop /usr/local/telescope/comm/Dome.out 5 "stop ←
     complete"
89 domepark=/usr/local/telescope/comm/Dome.in Az:2.7 /usr/local/telescope/comm/Dome.out 150 "←
     azimuth command complete" "error"
90 domeauto_on=/usr/local/telescope/comm/Dome.in auto /usr/local/telescope/comm/Dome.out
91 domeauto_off=/usr/local/telescope/comm/Dome.in off /usr/local/telescope/comm/Dome.out
92 domestatus=/usr/local/telescope/comm/Dome.in status /usr/local/telescope/comm/Dome.out 3 "←
     Dome" "dome"
93
94 # Cover commands
95 coveropen=/usr/local/telescope/comm/Cover.in coveropen /usr/local/telescope/comm/Cover.out 60←
     "open complete"
96 coverclose=/usr/local/telescope/comm/Cover.in coverclose /usr/local/telescope/comm/Cover.out ←
     60 "close complete"
97 coverstatus=/usr/local/telescope/comm/Cover.in status /usr/local/telescope/comm/Cover.out 3

```

Código 37: Contenido del fichero talon_alias

12.4. Sistemas de meteorología

12.4.1. previstorm.c

```

1 #include <sys/fcntl.h>
2 #include <termios.h>
3 #include <unistd.h>
4 #include <stdint.h>
5 #include <string.h>
6 #include <stdio.h>
7 #include <math.h>
8 #include <unistd.h>
9 #include <signal.h>
10 #include <sys/ioctl.h>
11 #include <time.h>
12 #include <errno.h>
13 #include <sys/types.h>
14 #include <sys/stat.h>
15
16 // SERIAL CONFIGURATION
17 #define BAUDRATE B19200
18 int fd=-1;
19 struct termios oldtio,newtio;
20
21 // FUNCTION DEFINITION
22 void port_open(char *device);
23 int port_read(char *buffer,int len);
24 void port_close(void);
25 double read_time(void);
26 int read_data(double jd,char *outfile);
27 int read_number(char number);
28
29 // ALARM PROTECTION
30 void alarm_handler(int signum) {
31     port_close();
32 }
33
34 // FUNCTION IMPLEMENTATION
35 int port_read(char *buffer,int len) {
36     int retval;
37
38     signal(SIGALRM,alarm_handler);
39     alarm(5);

```

```

40     retval=read(fd,buffer,len);
41     if(retval== -1) {
42         //~ printf("ERROR: %s (%d)\n",strerror(errno),errno);
43         if(errno!=EAGAIN) port_close();
44         retval=0;
45     }
46     alarm(0);
47     return retval;
48 }
49
50 void port_open(char *device) {
51     fd=open(device,O_RDWR|O_NONBLOCK|O_NOCTTY);
52     if(fd== -1) return;
53     tcgetattr(fd,&oldtio);
54     bzero(&newtio,sizeof(newtio));
55     newtio.c_cflag=CS8|CLOCAL|CREAD;
56     newtio.c_iflag=IGNPAR;
57     newtio.c_oflag=0;
58     newtio.c_lflag=0;
59     newtio.c_cc[VTIME]=1;
60     newtio.c_cc[VMIN]=1;
61     tcflush(fd,TCIOFLUSH);
62     cfsetispeed(&newtio, BAUDRATE);
63     cfsetospeed(&newtio, BAUDRATE);
64     tcsetattr(fd,TCSANOW,&newtio);
65 }
66
67 void port_close(void) {
68     tcsetattr(fd,TCSANOW,&oldtio);
69     close(fd);
70     fd=-1;
71 }
72
73 double read_time(void) {
74     time_t t1;
75     struct tm *t2;
76     double HR, TH, TM, TS, DD, DY, MM, YY, GR, JD, JD2=-1;
77
78     t1=time(NULL);
79     t2=gmtime(&t1);
80     //printf("%s\n",asctime(t2));
81     TH=t2->tm_hour;
82     TM=t2->tm_min;
83     TS=t2->tm_sec;
84     DD=t2->tm_mday;
85     MM=t2->tm_mon+1;
86     YY=t2->tm_year+1900;
87     //printf("%f-%f-%f %f:%f:%f\n",YY,MM,DD,TH,TM,TS);
88     HR = TH + TM/60 + TS/3600;
89     DD = DD + HR/24;
90     DY = floor(DD);
91     if (MM<3) { YY = YY - 1; MM = MM + 12; }
92     if ((YY + MM / 100 + DY / 10000) >= 1582.1015) GR =2-floor(YY/100)+floor(floor(YY/100)/4)--;
93     ;
94     else GR = 0;
95     JD=floor(365.25* YY)+floor(30.6001*(MM+1))+DY+1720994.5+GR;
96     //printf("JD2=%f\n",JD2);
97     return JD2;
98 }
99
100 int read_number(char number) {
101     int value=0;
102     switch(number) {
103         case '0': value=0; break;
104         case '1': value=1; break;
105         case '2': value=2; break;
106         case '3': value=3; break;
107         case '4': value=4; break;
108         case '5': value=5; break;
109         case '6': value=6; break;
110         case '7': value=7; break;
111         case '8': value=8; break;
112         case '9': value=9; break;
113         case 'a': value=10; break;
114         case 'b': value=11; break;
115         case 'c': value=12; break;
116         case 'd': value=13; break;
117         case 'e': value=14; break;
118         case 'f': value=15; break;
119         case 'A': value=10; break;
120         case 'B': value=11; break;
121         case 'C': value=12; break;

```

```

122         case 'D': value=13; break;
123         case 'E': value=14; break;
124         case 'F': value=15; break;
125     }
126     return value;
127 }
128
129 int read_data(double jd,char *outfile) {
130     char buffer[100];
131     FILE *fp;
132     uint16_t tempval;
133     int retval;
134
135     // sprintf(buffer,"%s",outfile);
136     // tempval=read_number(buffer[4])+read_number(buffer[3])*16+read_number(buffer[2])*256+←
137     // read_number(buffer[1])*16*256;
138     // printf("%s => %f\n",outfile,((float)(tempval))/256);
139     // _exit(0);
140
141     retval=port_read(buffer,99);
142
143     // printf("fd=%d\n",fd);
144     // int i;
145     // for(i=0;i<11;i++) printf("0x%02X ",buffer[i]&0xFF);
146     // printf("\n");
147
148     if(fd!=-1 && retval==11 && buffer[0]==‘a’) {
149         // OPEN OUTPUT FILE
150         fp=fopen(outfile,"w");
151         // ESFIELD PREVISTORM
152         fprintf(fp,"ESFIELD\n");
153         fprintf(fp,"PREVISTORM\n");
154         tempval=read_number(buffer[4])+read_number(buffer[3])*16+read_number(buffer[2])*256+←
155             read_number(buffer[1])*16*256;
156         fprintf(fp,"%f\n",((float)(tempval))/256);
157         fprintf(fp,"%.5f\n",jd);
158         // CLOSE OUTPUT FILE
159         fclose(fp);
160         retval=1;
161     } else {
162         retval=0;
163     }
164     return retval;
165 }
166
167 int main(int argc, char **argv) {
168     double jd;
169     int count;
170
171     // jd=read_time();
172     // read_data(jd,argv[1]);
173
174     // ARGUMENTS CHECK
175     if(argc!=3) {
176         printf("%s device outfile\n",argv[0]);
177         return 0;
178     }
179     // MAIN LOOP
180     while(1) {
181         port_open(argv[1]);
182         // printf("open");
183         count=0;
184         while(count<5 && fd!=-1) {
185             jd=read_time();
186             if(fd==-1) break;
187             count=read_data(jd,argv[2])?0:count+1;
188             if(fd==-1) break;
189             usleep(50000);
190         }
191         port_close();
192         sleep(1);
193     }
194     // NEVER RETURN!!!
195     return 0;
196 }
```

Código 38: Contenido del fichero previstorm.c

12.4.2. davis.c

```

1 #include <sys/fcntl.h>
2 #include <termios.h>
3 #include <unistd.h>
4 #include <stdint.h>
5 #include <string.h>
6 #include <stdio.h>
7 #include <math.h>
8 #include <unistd.h>
9 #include <signal.h>
10 #include <sys/ioctl.h>
11 #include <time.h>
12 #include <errno.h>
13 #include <sys/types.h>
14 #include <sys/stat.h>
15
16 // SERIAL CONFIGURATION
17 #define BAUDRATE B19200
18 int fd=-1;
19 struct termios oldtio,newtio;
20
21 // CRC16 TABLE
22 unsigned short crc_table [] = {
23     0x0, 0x1021, 0x2042, 0x3063, 0x4084, 0x50a5, 0x60c6, 0x70e7,
24     0x8108, 0x9129, 0xa14a, 0xb16b, 0xc18c, 0xd1ad, 0xe1ce, 0xf1ef,
25     0x1231, 0x2210, 0x3273, 0x4225, 0x52b5, 0x6294, 0x72f7, 0x82d6,
26     0x9339, 0x8318, 0xb37b, 0xa35a, 0xd3bd, 0xc39c, 0xf3ff, 0xe3de,
27     0x2462, 0x3443, 0x420, 0x1401, 0x64e6, 0x74c7, 0x44a4, 0x5485,
28     0xa56a, 0xb54b, 0x8528, 0x9509, 0xe5ee, 0xf5cf, 0xc5ac, 0xd58d,
29     0x3653, 0x2672, 0x1611, 0x630, 0x76d7, 0x66f6, 0x5695, 0x46b4,
30     0xb75b, 0xa77a, 0x9719, 0x8738, 0xf7df, 0xe7fe, 0xd79d, 0xc7bc,
31     0x48c4, 0x58e5, 0x6886, 0x78a7, 0x840, 0x1861, 0x2802, 0x3823,
32     0xc9cc, 0xd9ed, 0xe98e, 0xf9af, 0x8948, 0x9969, 0xa90a, 0xb92b,
33     0x5af5, 0x4ad4, 0x7ab7, 0x6a96, 0x1a71, 0xa50, 0x3a33, 0x2a12,
34     0xdbfd, 0xcbdc, 0xfbff, 0xeb9e, 0x9b79, 0x8b58, 0xbb3b, 0xab1a,
35     0x6ca6, 0x7c87, 0x4ce4, 0x5cc5, 0x2c22, 0x3c03, 0xc60, 0x1c41,
36     0xedae, 0xfd8f, 0xcdcc, 0xddcd, 0xad2a, 0xbd0b, 0x8d68, 0x9d49,
37     0x7e97, 0x6eb6, 0x5ed5, 0x4ef4, 0x3e13, 0x2e32, 0x1e51, 0xe70,
38     0xff9f, 0xefbe, 0xdfdd, 0xcfcc, 0xfb1b, 0xaf3a, 0x9f59, 0x8f78,
39     0x9188, 0x81a9, 0xb1ca, 0xa1eb, 0xd10c, 0xc12d, 0xf14e, 0xe16f,
40     0x1080, 0xa1, 0x30c2, 0x20e3, 0x5004, 0x4025, 0x7046, 0x6067,
41     0x83b9, 0x9398, 0xa3fb, 0xb3da, 0xc33d, 0xd31c, 0xe37f, 0xf35e,
42     0x2b1, 0x1290, 0x22f3, 0x32d2, 0x4235, 0x5214, 0x6277, 0x7256,
43     0xb5ea, 0xa5cb, 0x95a8, 0x8589, 0xf56e, 0xe54f, 0xd52c, 0xc50d,
44     0x34e2, 0x24c3, 0x14a0, 0x481, 0x7466, 0x6447, 0x5424, 0x4405,
45     0xa7db, 0xb7fa, 0x8799, 0x97b8, 0xe75f, 0xf77e, 0xc71d, 0xd73c,
46     0x26d3, 0x36f2, 0x691, 0x16b0, 0x6657, 0x7676, 0x4615, 0x5634,
47     0xd94c, 0xc96d, 0xf90e, 0xe92f, 0x99c8, 0x89e9, 0xb98a, 0xa9ab,
48     0x5844, 0x4865, 0x7806, 0x6827, 0x18c0, 0x8e1, 0x3882, 0x28a3,
49     0xcb7d, 0xdb5c, 0xeb3f, 0xfb1e, 0x8bf9, 0x9bd8, 0xabbb, 0xbb9a,
50     0x4a75, 0x5a54, 0x6a37, 0x7a16, 0xaf1, 0x1ad0, 0x2ab3, 0x3a92,
51     0xfd2e, 0xed0f, 0xdd6c, 0xcd4d, 0xbdaa, 0xad8b, 0x9de8, 0x8dc9,
52     0x7c26, 0x6c07, 0x5c64, 0x4c45, 0x3ca2, 0x2c83, 0x1ce0, 0xcc1,
53     0xef1f, 0xff3e, 0xcf5d, 0xdf7c, 0xaf9b, 0xbfba, 0x8fd9, 0x9ff8,
54     0x6e17, 0x7e36, 0x4e55, 0x5e74, 0x2e93, 0x3eb2, 0xed1, 0x1ef0,
55 };
56
57 // FUNCTION DEFINITION
58 void port_open(char *device);
59 int port_read(char *buffer,int len);
60 void port_close(void);
61 double read_time(void);
62 int read_data(double jd,char *outfile);
63 void wake_console(void);
64 int send_time(void);
65
66 // ALARM PROTECTION
67 void alarm_handler(int signum) {
68     port_close();
69 }
70
71 // FUNCTION IMPLEMENTATION
72 int port_read(char *buffer,int len) {
73     int retval;
74
75     signal(SIGALRM,alarm_handler);
76     alarm(5);
77     retval=read(fd,buffer,len);
78     if(retval==-1) {
79         //~ printf("ERROR: %s (%d)\n",strerror(errno),errno);
80         if(errno!=EAGAIN) port_close();
81     }
82 }

```

```

81         retval=0;
82     }
83     alarm(0);
84     return retval;
85 }
86
87 void port_open(char *device) {
88     fd=open(device,O_RDWR|O_NONBLOCK|O_NOCTTY);
89     if(fd==-1) return;
90     tcgetattr(fd,&oldtio);
91     bzero(&newtio,sizeof(newtio));
92     newtio.c_cflag=CS8|CLOCAL|CREAD;
93     newtio.c_iflag=IGNPAR;
94     newtio.c_oflag=0;
95     newtio.c_lflag=0;
96     newtio.c_cc[VTIME]=1;
97     newtio.c_cc[VMIN]=1;
98     tcflush(fd,TCIOFLUSH);
99     cfsetispeed(&newtio, BAUDRATE);
100    cfsetospeed(&newtio, BAUDRATE);
101    tcsetattr(fd,TCSANOW,&newtio);
102 }
103
104 void port_close(void) {
105     tcsetattr(fd,TCSANOW,&oldtio);
106     close(fd);
107     fd=-1;
108 }
109
110 void wake_console(void) {
111     char buffer[3];
112     int retval=0;
113     int count=0;
114
115     while(count<50 && fd!=-1) {
116         tcflush(fd,TCIOFLUSH);
117         write(fd,"\\n",1);
118         usleep(100000);
119         retval=port_read(buffer,2);
120         if(fd==-1) break;
121         if(retval==2 && buffer[0]=='\\n' && buffer[1]=='\\r') break;
122         count++;
123     }
124     if(count>=50) port_close();
125 }
126
127 double read_time(void) {
128     char buffer[9];
129     int retval;
130     double HR, TH, TM, TS, DD, DY, MM, YY, GR, JD, JD2=-1;
131
132     write(fd,"GETTIME\\n",8);
133     usleep(100000);
134     retval=port_read(buffer,1);
135     if(fd!=-1 && retval==1 && buffer[0]==0x06) {
136         retval=port_read(buffer,8);
137         if(fd!=-1 && retval==8) {
138             //printf("%d-%d-%d %d:%d:%d\\n",buffer[5]+1900,buffer[4],buffer[3],buffer[2],←
139             //       buffer[1],buffer[0]);
140             TH=buffer[2];
141             TM=buffer[1];
142             TS=buffer[0];
143             DD=buffer[3];
144             MM=buffer[4];
145             YY=buffer[5]+1900;
146             //printf("%f-%f-%f %f:%f:%f\\n",YY,MM,DD,TH,TM,TS);
147             HR = TH + TM/60 + TS/3600;
148             DD = DD + HR/24;
149             DY = floor(DD);
150             if ((MM<3) { YY = YY - 1; MM = MM + 12; }
151             if ((YY + MM / 100 + DY / 10000) >= 1582.1015) GR =2-floor(YY/100)+floor(floor(YY←
152                                         /100)/4);
153             else GR = 0;
154             JD=floor(365.25* YY)+floor(30.6001*(MM+1))+DY+1720994.5+GR;
155             JD2=JD+HR/24;
156             //printf("JD2=%f\\n",JD2);
157         }
158     }
159     return JD2;
160 }
161
162 int read_data(double jd,char *outfile) {
163     char buffer[100];

```

```

162 FILE *fp;
163 uint16_t tempval;
164 int retval;
165
166 if(jd!=-1) {
167     tcflush(fd,TCIOFLUSH);
168     write(fd,"LOOP 1\n",7);
169     usleep(100000);
170     retval=port_read(buffer,1);
171     //printf("0x%02X ",buffer[0]&0xFF);
172     if(fd!=-1 && retval==1 && buffer[0]==0x06) {
173         retval=port_read(buffer,99);
174         //int i;
175         //for(i=0;i<100;i++) printf("0x%02X ",buffer[i]&0xFF);
176         //printf("\n");
177         if(fd!=-1 && retval==99 && buffer[0]=='L' && buffer[1]=='0' && buffer[2]=='0'){
178             // OPEN OUTPUT FILE
179             fp=fopen(outfile,"w");
180             // TEMPERATURE DAVIS-INDOOR
181             fprintf(fp,"TEMPERATURE\n");
182             fprintf(fp,"DAVIS-INDOOR\n");
183             memcpy(&tempval,&buffer[9],sizeof(uint16_t));
184             fprintf(fp,"%1f\n",((tempval/10.0)-32)/1.8);
185             fprintf(fp,"%5f\n",jd);
186             // HUMIDITY DAVIS-INDOOR
187             fprintf(fp,"HUMIDITY\n");
188             fprintf(fp,"DAVIS-INDOOR\n");
189             fprintf(fp,"%d\n",buffer[11]);
190             fprintf(fp,"%5f\n",jd);
191             // TEMPERATURE DAVIS-TJO
192             fprintf(fp,"TEMPERATURE\n");
193             fprintf(fp,"DAVIS-TJO\n");
194             fprintf(fp,"%1f\n",((unsigned char)(buffer[18]-90)-32)/1.8);
195             fprintf(fp,"%5f\n",jd);
196             // HUMIDITY DAVIS-TJO
197             fprintf(fp,"HUMIDITY\n");
198             fprintf(fp,"DAVIS-TJO\n");
199             fprintf(fp,"%d\n",(unsigned char)buffer[34]);
200             fprintf(fp,"%5f\n",jd);
201             // TEMPERATURE DAVIS-TOWER
202             fprintf(fp,"TEMPERATURE\n");
203             fprintf(fp,"DAVIS-TOWER\n");
204             memcpy(&tempval,&buffer[12],sizeof(uint16_t));
205             fprintf(fp,"%1f\n",((tempval/10.0)-32)/1.8);
206             fprintf(fp,"%5f\n",jd);
207             // HUMIDITY DAVIS-TOWER
208             fprintf(fp,"HUMIDITY\n");
209             fprintf(fp,"DAVIS-TOWER\n");
210             fprintf(fp,"%d\n",buffer[33]);
211             fprintf(fp,"%5f\n",jd);
212             // WINDSPEED DAVIS-TOWER
213             fprintf(fp,"WINDSPEED\n");
214             fprintf(fp,"DAVIS-TOWER\n");
215             fprintf(fp,"%0.1f\n",buffer[14]*0.44704);
216             fprintf(fp,"%5f\n",jd);
217             // WINDDIR DAVIS-TOWER
218             fprintf(fp,"WINDDIR\n");
219             fprintf(fp,"DAVIS-TOWER\n");
220             memcpy(&tempval,&buffer[16],sizeof(uint16_t));
221             fprintf(fp,"%d\n",tempval);
222             fprintf(fp,"%5f\n",jd);
223             // PRESSURE DAVIS-TOWER
224             fprintf(fp,"PRESSURE\n");
225             fprintf(fp,"DAVIS-TOWER\n");
226             memcpy(&tempval,&buffer[7],sizeof(uint16_t));
227             fprintf(fp,"%3f\n",tempval*33.86/1000);
228             fprintf(fp,"%5f\n",jd);
229             // RAIN DAVIS-TOWER
230             fprintf(fp,"RAIN\n");
231             fprintf(fp,"DAVIS-TOWER\n");
232             memcpy(&tempval,&buffer[50],sizeof(uint16_t));
233             fprintf(fp,"%0.1f\n",tempval*0.254);
234             fprintf(fp,"%5f\n",jd);
235             // TEMPERATURE DAVIS-ARES
236             fprintf(fp,"TEMPERATURE\n");
237             fprintf(fp,"DAVIS-ARES\n");
238             fprintf(fp,"%1f\n",((unsigned char)(buffer[19]-90)-32)/1.8);
239             fprintf(fp,"%5f\n",jd);
240             // HUMIDITY DAVIS-ARES
241             fprintf(fp,"HUMIDITY\n");
242             fprintf(fp,"DAVIS-ARES\n");
243             fprintf(fp,"%d\n",(unsigned char)buffer[35]);
244             fprintf(fp,"%5f\n",jd);

```

```

245         // CLOSE OUTPUT FILE
246         fclose(fp);
247         retval=1;
248     } else {
249         retval=0;
250     }
251 } else {
252     retval=0;
253 }
254 } else {
255     retval=0;
256 }
257 return retval;
258 }
259
260 int send_time(void) {
261     time_t t1;
262     struct tm *t2;
263     char buffer[9];
264     int retval;
265     unsigned short crc;
266
267     write(fd,"SETTIME\n",8);
268     usleep(100000);
269     retval=port_read(buffer,1);
270     if(fd!=-1 && retval==1 && buffer[0]==0x06) {
271         t1=time(NULL);
272         t2=gmtime(&t1);
273         //printf("%s\n",asctime(t2));
274         buffer[0]=t2->tm_sec;
275         buffer[1]=t2->tm_min;
276         buffer[2]=t2->tm_hour;
277         buffer[3]=t2->tm_mday;
278         buffer[4]=t2->tm_mon+1;
279         buffer[5]=t2->tm_year;
280         crc=0;
281         crc=crc_table[(crc>>8)^((unsigned char)buffer[0])^(crc<<8)];
282         crc=crc_table[(crc>>8)^((unsigned char)buffer[1])^(crc<<8)];
283         crc=crc_table[(crc>>8)^((unsigned char)buffer[2])^(crc<<8)];
284         crc=crc_table[(crc>>8)^((unsigned char)buffer[3])^(crc<<8)];
285         crc=crc_table[(crc>>8)^((unsigned char)buffer[4])^(crc<<8)];
286         crc=crc_table[(crc>>8)^((unsigned char)buffer[5])^(crc<<8)];
287         buffer[6]=(unsigned char)((crc>>8)&0xff);
288         buffer[7]=(unsigned char)(crc&0xff);
289         //~ int i;
290         //~ for(i=0;i<8;i++) printf("0x%02X ",buffer[i]&0xFF);
291         //~ printf("\n");
292         write(fd,buffer,8);
293         usleep(100000);
294         retval=port_read(buffer,1);
295         if(fd!=-1 && retval==1 && buffer[0]==0x06) {
296             //printf("SETTIME OK\n");
297             retval=1;
298         } else {
299             //printf("SETTIME KO (0x%02X)\n",buffer[0]);
300             retval=0;
301         }
302     } else {
303         retval=0;
304     }
305     return retval;
306 }
307
308 int main(int argc, char **argv) {
309     double jd;
310     int count;
311     int count2;
312
313     // ARGUMENTS CHECK
314     if(argc!=3) {
315         printf("%s device outfile\n",argv[0]);
316         return 0;
317     }
318     // MAIN LOOP
319     while(1) {
320         port_open(argv[1]);
321         count=0;
322         while(count<5 && fd!=-1) {
323             count=0;
324             count2=0;
325             while(count<5 && fd!=-1 && count2<6000) {
326                 wake_console();
327                 if(fd===-1) break;

```

```

328         jd=read_time();
329         if(fd== -1) break;
330         wake_console();
331         if(fd== -1) break;
332         count=read_data(jd,argv[2])?0:count+1;
333         if(fd== -1) break;
334         usleep(100000);
335         count2++;
336     }
337     count=0;
338     while(count<5 && fd!= -1) {
339         wake_console();
340         if(fd== -1) break;
341         count=send_time()?0:count+1;
342         if(fd== -1) break;
343         if(count==0) break;
344         usleep(100000);
345     }
346     port_close();
347     sleep(1);
348 }
349 // NEVER RETURN!!!
350 return 0;
351 }
```

Código 39: Contenido del fichero davis.c

12.4.3. emsmc.php

```

1  #!/usr/bin/php -q
2  <?php
3  ini_set("date.timezone","Europe/Madrid");
4  // DEFAULT VAULES
5  $command="/home/optjo/gitrepos/tjo/rebei/pbcdl_comm -c /home/optjo/gitrepos/tjo/rebei/EMSMC.<-
       xml";
6  $input="/home/optjo/EMSMC/.working/Public.tmp";
7  $output="/tmp/emsmc.txt";
8  $delay=1;
9  $echo=1;
10 // PROCESS ARGUMENTS
11 array_shift($argv);
12 $param=array_shift($argv);
13 while($param) {
14     switch($param) {
15         case "-c":
16             $command=array_shift($argv);
17             break;
18         case "-i":
19             $input=array_shift($argv);
20             break;
21         case "-o":
22             $output=array_shift($argv);
23             break;
24         case "-d":
25             $delay=array_shift($argv);
26             break;
27         case "-e":
28             $echo=array_shift($argv);
29             break;
30         default:
31             die("Unknown argument $param\n");
32             break;
33     }
34     $param=array_shift($argv);
35 }
36 // HELPER FUNCTIONS
37 function _csv_parser($line) {
38     // SPECIAL CASE
39     if(trim($line)== "") return array();
40     $line=explode(",",$line);
41     foreach($line as $key=>$val) $line[$key]=trim($val);
42     foreach($line as $key=>$val) if(substr($val,0,1)== ',') $line[$key]=substr($val,1);
43     foreach($line as $key=>$val) if(substr($val,-1,1)== ',') $line[$key]=substr($val,0,-1);
44     // TRICK TO FIX AN UNUSUAL BUG
45     foreach($line as $key=>$val) $line[$key]=str_replace(',,',",",$val);
```

```

46 $line=implode(",",$line);
47 $line=explode(",",$line);
48 foreach($line as $key=>$val) $line[$key]=trim($val);
49 foreach($line as $key=>$val) if(substr($val,0,1)==',') $line[$key]=substr($val,1);
50 foreach($line as $key=>$val) if(substr($val,-1,1)==',') $line[$key]=substr($val,0,-1);
51 // CONTINUE
52 return $line;
53 }
54 // MAIN LOOP, HAVE FUN
55 while(1) {
56     // EXECUTE THE COMMAND
57     exec($command);
58     // OPEN INPUT FILE
59     $fp=fopen($input,"r");
60     if($fp) {
61         // READ DATA
62         $info=__csv_parser(fgets($fp));
63         $fields=__csv_parser(fgets($fp));
64         $units=__csv_parser(fgets($fp));
65         $data=array();
66         $temp=__csv_parser(fgets($fp));
67         if(count($temp)) $data=$temp;
68         while(!feof($fp)) {
69             $temp=__csv_parser(fgets($fp));
70             if(count($temp)) $data[]=$temp;
71         }
72         fclose($fp);
73         // CHECK RESULTS
74         $count1=count($fields);
75         $count2=count($units);
76         $count3=count($data);
77         if($count1==$count2 && $count2==$count3) {
78             // PREPARE TIMESTAMP
79             $temp=strtok($data[0],".");
80             $now=strtotime($temp);
81             $month=date("m",$now);
82             $day=date("d",$now);
83             $year=date("Y",$now);
84             $hour=date("H",$now);
85             $minute=date("i",$now);
86             $second=date("s",$now);
87             $jd=gregoriantojd($month,$day,$year)+((((($hour-12)*3600+$minute*60+$second)←
88             %86400)/86400);
89             // DUMP OUTPUT
90             ob_start();
91             // TEMPERATURE SMC
92             echo "TEMPERATURE\n";
93             echo "SMC\n";
94             echo $data[2]."\n";
95             echo "$jd\n";
96             // HUMIDITY SMC
97             echo "HUMIDITY\n";
98             echo "SMC\n";
99             echo $data[3]."\n";
100            echo "$jd\n";
101            // WINDSPEED SMC
102            echo "WINDSPEED\n";
103            echo "SMC\n";
104            echo $data[15]."\n";
105            echo "$jd\n";
106            // PRESSURE SMC
107            echo "PRESSURE\n";
108            echo "SMC\n";
109            echo $data[6]."\n";
110            echo "$jd\n";
111            // SOLARRAD SMC
112            echo "SOLARRAD\n";
113            echo "SMC\n";
114            echo $data[4]."\n";
115            echo "$jd\n";
116            // DUMP OF ALL DATA
117            for($i=0;$i<$count1;$i++) {
118                if($units[$i]) {
119                    echo $fields[$i]."\n";
120                    echo $units[$i]."\n";
121                    echo $data[$i]."\n";
122                }
123                $buffer=ob_get_clean();
124                file_put_contents($output,$buffer);
125                if($echo) echo $buffer;
126            } else {
127                echo "Skipping invalid data from $input\n";

```

```

128         }
129     } else {
130         echo "Could not read input from $input\n";
131     }
132     sleep($delay);
133 }
134 ?>

```

Código 40: Contenido del fichero emsmc.php

12.4.4. EMSMC.xml

```

1  <?xml version="1.0" encoding="utf-8" ?>
2  <!-- CONFIG FILE TO SETUP DATA COLLECTION FROM PAKBUS LOGGER -->
3  <!-- Comments with "##" will need to be set for deployment -->
4  <!-- ** Set the station name to a brief description. -->
5  <COLLECTION logger="" station_name="">
6      <!-- Serial Port configuration -->
7      <CONNECTION type="serial">
8          <!-- ** Set the port_name to the serial port address -->
9          <port_name>/dev/emsmc</port_name>
10         <!-- ** Set the baud rate, if different -->
11         <baud_rate>38400</baud_rate>
12         <vtimer>10</vtimer>
13     </CONNECTION>
14     <!-- Turn on debugging : use TRUE/FALSE -->
15     <DEBUG>FALSE</DEBUG>
16     <!-- Data output options -->
17     <DATA>
18         <WORKING_PATH>/home/optjo/EMSMC/</WORKING_PATH>
19         <!--
20             Add a table entry to collect data from it. The
21             file_span_secs parameter can be used to set file
22             durations. sample_int_secs parameter indicates the
23             update frequency, which is used to determine if a data
24             file is complete and be moved up from the
25             ../../working directory.
26             -->
27         <COLLECT_TABLE>
28             <TABLE>Public</TABLE>
29             <!-- <TABLE>Status</TABLE> -->
30             <!-- <TABLE>Tablai</TABLE> -->
31             <!-- <TABLE>Tabla2</TABLE> -->
32         </COLLECT_TABLE>
33     </DATA>
34     <!-- PakBus protocol settings -->
35     <!-- The following are default values -->
36     <PAKBUS>
37         <DST_PAKBUS_ID>1</DST_PAKBUS_ID>
38         <DST_NODE_PAKBUS_ID>1</DST_NODE_PAKBUS_ID>
39         <!-- Default value for security code is zero -->
40         <SECURITY_CODE>0</SECURITY_CODE>
41     </PAKBUS>
42 </COLLECTION>

```

Código 41: Contenido del fichero EMSMC.xml

12.4.5. vaisala.php

```

1  #!/usr/bin/php -q
2  <?php
3  ini_set("date.timezone", "Europe/Madrid");
4  // DEFAULT VAULES
5  $command="indi_getprop";
6  $output="/tmp/vaisala.txt";
7  $host="AA.BB.CC.DD";
8  $port=XXXX;
9  $delay=1;
10 $echo=1;

```

```

11 // PROCESS ARGUMENTS
12 array_shift($argv);
13 $param=array_shift($argv);
14 while($param) {
15     switch($param) {
16         case "-o":
17             $output=array_shift($argv);
18             break;
19         case "-h":
20             $host=array_shift($argv);
21             break;
22         case "-p":
23             $port=array_shift($argv);
24             break;
25         case "-d":
26             $delay=array_shift($argv);
27             break;
28         case "-e":
29             $echo=array_shift($argv);
30             break;
31         default:
32             die("Unknown argument $param\n");
33             break;
34     }
35     $param=array_shift($argv);
36 }
37 // MAIN LOOP, HAVE FUN
38 while(1) {
39     // RETRIEVE ALL DATA
40     $lista=array(
41         "Environment.Now.AirTemp",
42         "Environment.Now.DewPoint",
43         "Environment.Now.WindChill",
44         "Environment.Now.AirPressure",
45         "Environment.Now.Humidity",
46         "Environment.Now.WindDir",
47         "Environment.Now.WindSpeed",
48         "Environment.Now.WindGust",
49         "Environment.Now.RainAccum",
50         "Environment.Now.RainDetected"
51     );
52     $datas=array();
53     foreach($lista as $key=>$val) {
54         ob_start();
55         passthru("${command} -p ${port} -h ${host} ${val}");
56         $buffer=ob_get_clean();
57         $buffer=trim($buffer);
58         $temp=explode("=", $buffer);
59         if(count($temp)==2) {
60             $temp2=explode(".", $temp[0]);
61             if(count($temp2)==3) {
62                 $key=strtoupper(trim($temp2[2]));
63                 $val=trim($temp[1]);
64                 $datas[$key]=$val;
65             }
66         }
67     }
68     // CHECK RESULTS
69     $count1=count($lista);
70     $count2=count($datas);
71     if($count1==$count2) {
72         // PREPARE TIMESTAMP
73         $now=time();
74         $month=date("m", $now);
75         $day=date("d", $now);
76         $year=date("Y", $now);
77         $hour=date("H", $now);
78         $minute=date("i", $now);
79         $second=date("s", $now);
80         $jd=gregoriantojd($month, $day, $year)+(((($hour-12)*3600+$minute*60+$second)%86400)←
81         /86400);
82         // DUMP OUTPUT
83         ob_start();
84         foreach($datas as $key=>$val) {
85             echo "$key\n";
86             echo "VAISALA\n";
87             echo "$val\n";
88             echo "$jd\n";
89         }
90         $buffer=ob_get_clean();
91         file_put_contents($output, $buffer);
92         if($echo) echo $buffer;
93     }

```

```

93     sleep($delay);
94 }
95 ?>

```

Código 42: Contenido del fichero vaisala.php

12.4.6. raindetect.c

```

1 #include <sys/fcntl.h>
2 #include <termios.h>
3 #include <unistd.h>
4 #include <stdint.h>
5 #include <string.h>
6 #include <stdio.h>
7 #include <math.h>
8 #include <unistd.h>
9 #include <signal.h>
10 #include <sys/ioctl.h>
11 #include <time.h>
12 #include <errno.h>
13 #include <sys/types.h>
14 #include <sys/stat.h>
15
16 // SERIAL CONFIGURATION
17 int fd=-1;
18
19 // FUNCTION DEFINITION
20 void port_open(char *device);
21 void port_close(void);
22 double read_time(void);
23 int read_data(double jd,char *outfile);
24
25 // FUNCTION IMPLEMENTATION
26 void port_open(char *device) {
27     fd=open(device,O_RDWR|O_NOCTTY);
28 }
29
30 void port_close(void) {
31     close(fd);
32     fd=-1;
33 }
34
35 double read_time(void) {
36     time_t t1;
37     struct tm *t2;
38     double HR, TH, TM, TS, DD, DY, MM, YY, GR, JD, JD2=-1;
39
40     t1=time(NULL);
41     t2=gmtime(&t1);
42     //printf("%s\n",asctime(t2));
43     TH=t2->tm_hour;
44     TM=t2->tm_min;
45     TS=t2->tm_sec;
46     DD=t2->tm_mday;
47     MM=t2->tm_mon+1;
48     YY=t2->tm_year+1900;
49     //printf("%f-%f-%f %f:%f:%f\n",YY,MM,DD,TH,TM,TS);
50     HR = TH + TM/60 + TS/3600;
51     DD = DD + HR/24;
52     DY = floor(DD);
53     if (MM<3) { YY = YY - 1; MM = MM + 12; }
54     if ((YY + MM / 100 + DY / 10000) >= 1582.1015) GR =2-floor(YY/100)+floor(YY/100)/4
55     ;
56     else GR = 0;
57     JD=floor(365.25* YY)+floor(30.6001*(MM+1))+DY+1720994.5+GR;
58     //printf("JD2=%f\n",JD2);
59     return JD2;
60 }
61
62 int read_data(double jd,char *outfile) {
63     FILE *fp;
64     int retval=0;
65
66     if(fd>=0) {
67         if(ioctl(fd,TIOCGET,&retval)==0) {

```

```

68          // OPEN OUTPUT FILE
69          fp=fopen(outfile,"w");
70          // RAINDETECT RAINDETECTOR
71          fprintf(fp,"RAINDETECT\n");
72          fprintf(fp,"RAINDETECTOR\n");
73          fprintf(fp,"%s\n", (retval&TIOCMB_DSR)>0?"TRUE":"FALSE");
74          fprintf(fp,"%f\n",jd);
75          // CLOSE OUTPUT FILE
76          fclose(fp);
77          retval=1;
78      }
79  }
80  return retval;
81 }
82
83 int main(int argc, char **argv) {
84     int fd;
85     double jd;
86     int count;
87
88     // ARGUMENTS CHECK
89     if(argc!=3) {
90         printf("%s device outfile\n", argv[0]);
91         return 0;
92     }
93     // MAIN LOOP
94     while(1) {
95         port_open(argv[1]);
96         count=0;
97         while(count<5 && fd!=-1) {
98             jd=read_time();
99             if(fd==-1) break;
100            count=read_data(jd, argv[2])?0:count+1;
101            if(fd==-1) break;
102            usleep(100000);
103        }
104        port_close();
105        sleep(1);
106    }
107    // NEVER RETURN!!!
108    return 0;
109 }
```

Código 43: Contenido del fichero raindetect.c

12.4.7. cloudsensor.c

```

1 #include <sys/fcntl.h>
2 #include <termios.h>
3 #include <unistd.h>
4 #include <stdint.h>
5 #include <string.h>
6 #include <stdio.h>
7 #include <math.h>
8 #include <unistd.h>
9 #include <signal.h>
10 #include <sys/ioctl.h>
11 #include <time.h>
12 #include <errno.h>
13 #include <sys/types.h>
14 #include <sys/stat.h>
15
16 // SERIAL CONFIGURATION
17 #define BAUDRATE B4800
18 int fd=-1;
19 struct termios oldtio,newtio;
20
21 // FUNCTION DEFINITION
22 void port_open(char *device);
23 int port_read(char *buffer,int len);
24 void port_close(void);
25 double read_time(void);
26 int read_data(double jd,char *outfile);
27
28 // ALARM PROTECTION
29 void alarm_handler(int signum) {
```

```

30     port_close();
31 }
32
33 // FUNCTION IMPLEMENTATION
34 int port_read(char *buffer,int len) {
35     int retval;
36
37     signal(SIGALRM,alarm_handler);
38     alarm(5);
39     retval=read(fd,buffer,len);
40     if(retval== -1) {
41         //~ printf("ERROR: %s (%d)\n",strerror(errno),errno);
42         if(errno!=EAGAIN) port_close();
43         retval=0;
44     }
45     alarm(0);
46     return retval;
47 }
48
49 void port_open(char *device) {
50     //fd=open(device,O_RDWR|O_NONBLOCK|O_NOCTTY);
51     fd=open(device,O_RDWR|O_NOCTTY);
52     if(fd== -1) return;
53     tcgetattr(fd,&oldtio);
54     bzero(&newtio,sizeof(newtio));
55     newtio.c_cflag=CS8|CLOCAL|CREAD;
56     newtio.c_iflag=IGNPAR;
57     newtio.c_oflag=0;
58     newtio.c_lflag=0;
59     newtio.c_cc[VTIME]=1;
60     newtio.c_cc[VMIN]=1;
61     tcflush(fd,TCIOFLUSH);
62     cfsetispeed(&newtio, BAUDRATE);
63     cfsetospeed(&newtio, BAUDRATE);
64     tcsetattr(fd,TCSANOW,&newtio);
65 }
66
67 void port_close(void) {
68     tcsetattr(fd,TCSANOW,&oldtio);
69     close(fd);
70     fd=-1;
71 }
72
73 double read_time(void) {
74     time_t t1;
75     struct tm *t2;
76     double HR, TH, TM, TS, DD, DY, MM, YY, GR, JD, JD2=-1;
77
78     t1=time(NULL);
79     t2=gmtime(&t1);
80     //printf("%s\n",asctime(t2));
81     TH=t2->tm_hour;
82     TM=t2->tm_min;
83     TS=t2->tm_sec;
84     DD=t2->tm_mday;
85     MM=t2->tm_mon+1;
86     YY=t2->tm_year+1900;
87     //printf("%f-%f-%f %f:%f:%f\n",YY,MM,DD,TH,TM,TS);
88     HR = TH + TM/60 + TS/3600;
89     DD = DD + HR/24;
90     DY = floor(DD);
91     if (MM<3) { YY = YY - 1; MM = MM + 12; }
92     if ((YY + MM / 100 + DY / 10000) >= 1582.1015) GR =2-floor(YY/100)+floor(floor(YY/100)/4)--;
93     ;
94     else GR = 0;
95     JD=floor(365.25* YY)+floor(30.6001*(MM+1))+DY+1720994.5+GR;
96     JD2=JD+HR/24;
97     //printf("JD2=%f\n",JD2);
98     return JD2;
99 }
100 int read_data(double jd,char *outfile) {
101     char buffer[1024];
102     FILE *fp;
103     uint16_t tempval;
104     int retval;
105
106     tcflush(fd,TCIOFLUSH);
107     buffer[0]=0x02;
108     buffer[1]='a';
109     buffer[2]='\n';
110     buffer[3]=0x01;
111     write(fd,buffer,4);

```

```

112     usleep(100000);
113     retval=port_read(buffer,1);
114     if(fd!=-1 && retval==1 && buffer[0]==0x02) {
115         usleep(100000);
116         retval=port_read(buffer,2);
117         if(fd!=-1 && retval==2 && buffer[0]=='M' && buffer[1]=='D') {
118             usleep(100000);
119             retval+=port_read(&buffer[retval],128);
120             usleep(100000);
121             retval+=port_read(&buffer[retval],128);
122             usleep(100000);
123             retval+=port_read(&buffer[retval],128);
124             usleep(100000);
125             retval+=port_read(&buffer[retval],128);
126             /* 1: humidstatTempCode: humidity and ambient temperature sensor
127             *    0 = OK.
128             *    1 = write failure for humidity.
129             *    2 = measurement never finished for humidity.
130             *    3 = write failure for ambient.
131             *    4 = measurement never finished for ambient.
132             *    5 = data line was not high for humidity,
133             *    6 = data line was not high for ambient.
134             */
135             unsigned int humidstatTempCode;
136             /* 2: cloudCond (dT = skyMinusAmbientTemperature):
137             *    0 = unknown (e.g., sensor wet, still heating up, etc; see field 9)
138             *    1 = clear (dt > eCloudyThresh)
139             *    2 = cloudy (dT <= eCloudyThresh)
140             *    3 = very cloudy (dT <= eEveryCloudyThresh)
141             */
142             unsigned int cloudCond;
143             /* 3: windCond:
144             *    0 = unknown (e.g., sensor wet)
145             *    1 = ok (windSpeed < eWindyThresh)
146             *    2 = windy (windSpeed >= eWindyThresh)
147             *    3 = very windy (windSpeed >= eEveryWindyThresh)
148             */
149             unsigned int windCond;
150             /* 4: rainCond:
151             *    0 = unknown
152             *    1 = not raining
153             *    2 = recently raining
154             *    3 = raining
155             */
156             unsigned int rainCond;
157             /* 5: skyCond (for cloud Sensor I programs)
158             *    0 unknown
159             *    1 clear
160             *    2 cloudy
161             *    3 very cloudy
162             *    4 wet
163             */
164             unsigned int skyCond;
165             /* 6: roofCloseRequested: 0 normally, 1 if roof close
166             * was requested on this cycle
167             */
168             unsigned int roofCloseRequested;
169             /* 7: sky-ambient: degrees C; 999.9 saturated hot, -999.9 saturated cold;
170             * -998.0 if sensor is wet.
171             */
172             double skyMinusAmbientTemperature;
173             /* 8: ambientTemperature: degrees C; if -40 and humidity is 0
174             * there is a problem with communication to those sensors
175             * (likely water where it shouldnt be).
176             */
177             double ambientTemperature;
178             /* 9: wind speed in km/hr, or:
179             *    -1. if still heating up,
180             *    -2. if wet,
181             *    -3. if the A/D from the wind probe is bad,
182             *    -4. if the probe is not heating.
183             */
184             double windSpeed;
185             /* 10: wetSensor: 'N' when dry, 'W' when wet now, 'w' wet in
186             * last minute.
187             */
188             char wetSensor;
189             /* 11: rainSensor: 'N' when no rain, 'R' when rain drops hit
190             * on this cycle, 'r' for drops in last minute.
191             */
192             char rainSensor;
193             /* 12: relative humidity in % (range 0..100); see comment
194             * about ambientTemperature.

```

```

195      */
196      int relativeHumidityPercentage;
197      /* 13: dewPointTemperature: in degrees C */
198      double dewPointTemperature;
199      /* 14: thermopile case temperature, 999.9 saturated hot,
200      * -999.9 saturated cold.
201      */
202      double caseTemperature;
203      /* 15: */
204      int rainHeaterPercentage;
205      /* 16: calibration black body temperature (factory only),
206      * 999.9 saturated hot, -99.9 saturated cold.
207      */
208      double blackBodyTemperature;
209      /* 17: rainHeaterState:
210      * 0 if too hot,
211      * 1 if at or nearly at requested temp.,
212      * 2..5 if too cold,
213      * 6 if cannot control due to a saturated case temperature
214      * (causes shutdown).
215      * 7 is used by firmware (tmain) to indicate that normal control
216      * is being used instead of direct use of this.
217      */
218      int rainHeaterState;
219      /* 18: voltage actually on the +24V at the sensor head */
220      double powerVoltage;
221      /* 19: anemometer tip temperature difference from ambient,
222      * limited by reducing anemometer heater power when 25C
223      * is reached.
224      */
225      double anemometerTemeratureDiff;
226      /* 20: maximum drop in wetness oscillator counts this cycle
227      * due to rain drops
228      */
229      int wetnessDrop;
230      /* 21: wetness oscillator count difference from base dry value. */
231      int wetnessAvg;
232      /* 22: wetness oscillator count difference for current dry from
233      * base dry value
234      */
235      int wetnessDry;
236      /* 23: rain heater PWM value */
237      int rainHeaterPWM;
238      /* 24: anemometer heater PWM value */
239      int anemometerHeaterPWM;
240      /* 25: thermopile raw A/D output */
241      unsigned int thermopileADC;
242      /* 26: thermopile thermistor raw A/D output */
243      unsigned int thermistorADC;
244      /* 27: power supply voltage monitor raw A/D output */
245      unsigned int powerADC;
246      /* 28: calibration block thermistor raw A/D output */
247      unsigned int blockADC;
248      /* 29: anemometer tip thermistor raw A/D output */
249      unsigned int anemometerThermistorADC;
250      /* 30: Davis vane raw A/D output (only for factory calibration). */
251      unsigned int davisVaneADC;
252      /* 31: external anemometer used (only for factory calibration) */
253      double dkMPH;
254      /* 32: external anemometer wind direction (only for factory calibration) */
255      int extAnemometerDirection;
256      /* 33: raw counts from the wetness oscillator */
257      int rawWetness0sc;
258      /* 34: DayCond value:
259      * 3 full daylight
260      * 2 twilight
261      * 1 night
262      * 0 unknown
263      */
264      unsigned int dayCond;
265      /* 35: daylight photodiode raw A/D output
266      * (0 means no light, 1023 max light).
267      */
268      unsigned int daylightADC;
269      // READ USING SSCANF
270      unsigned int nChars;
271      retval=sscanf(&buffer[1],"D %u %u %u %u %u %u %lf %lf %c %c %d %lf %lf %d %lf<-
272      %d %lf %lf %d %d %d %u %n",
273      &humidstatTempCode, &cloudCond,
274      &windCond, &rainCond, &skyCond, &roofCloseRequested,
275      &skyMinusAmbientTemperature, &ambientTemperature, &windSpeed,
276      &wetSensor, &rainSensor, &relativeHumidityPercentage,
277      &dewPointTemperature, &caseTemperature, &rainHeaterPercentage,

```

```

277     &blackBodyTemperature, &rainHeaterState, &powerVoltage,
278     &anemometerTemeratureDiff, &wetnessDrop, &wetnessAvg,
279     &wetnessDry, &rainHeaterPWM, &anemometerHeaterPWM,
280     &thermopileADC, &thermistorADC, &powerADC, &blockADC,
281     &anemometerThermistorADC, &davisVaneADC, &dkMPH,
282     &extAnemometerDirection, &rawWetnessOsc, &dayCond,
283     &daylightADC,
284     &nChars);
285     // printf("sscanf=%d\n",retval);
286     if(retval==35) {
287         // OPEN OUTPUT FILE
288         fp=fopen(outfile,"w");
289         // HUMIDSTATTEMPCODE CLOUD-SENSOR
290         fprintf(fp,"HUMIDSTATTEMPCODE\n");
291         fprintf(fp,"CLOUD-SENSOR\n");
292         switch(humidstatTempCode) {
293             case 0: sprintf(buffer,"OK"); break;
294             case 1: sprintf(buffer,"write failure for humidity"); break;
295             case 2: sprintf(buffer,"measurement never finished for humidity"); break;
296             case 3: sprintf(buffer,"write failure for ambient"); break;
297             case 4: sprintf(buffer,"measurement never finished for ambient"); break;
298             case 5: sprintf(buffer,"data line was not high for humidity"); break;
299             case 6: sprintf(buffer,"data line was not high for ambient"); break;
300             default: sprintf(buffer,"unknown"); break;
301         }
302         fprintf(fp,"%u (%s)\n",humidstatTempCode,buffer);
303         fprintf(fp,"%f\n",jd);
304         // CLOUDCOND CLOUD-SENSOR
305         fprintf(fp,"CLOUDCOND\n");
306         fprintf(fp,"CLOUD-SENSOR\n");
307         switch(cloudCond) {
308             case 0: sprintf(buffer,"unknown"); break;
309             case 1: sprintf(buffer,"clear"); break;
310             case 2: sprintf(buffer,"cloudy"); break;
311             case 3: sprintf(buffer,"very cloudy"); break;
312             default: sprintf(buffer,"unknown"); break;
313         }
314         fprintf(fp,"%u (%s)\n",cloudCond,buffer);
315         fprintf(fp,"%f\n",jd);
316         // WINDCOND CLOUD-SENSOR
317         fprintf(fp,"WINDCOND\n");
318         fprintf(fp,"CLOUD-SENSOR\n");
319         switch(windCond) {
320             case 0: sprintf(buffer,"unknown"); break;
321             case 1: sprintf(buffer,"ok"); break;
322             case 2: sprintf(buffer,"windy"); break;
323             case 3: sprintf(buffer,"very windy"); break;
324             default: sprintf(buffer,"unknown"); break;
325         }
326         fprintf(fp,"%u (%s)\n",windCond,buffer);
327         fprintf(fp,"%f\n",jd);
328         // RAINCOND CLOUD-SENSOR
329         fprintf(fp,"RAINCOND\n");
330         fprintf(fp,"CLOUD-SENSOR\n");
331         switch(rainCond) {
332             case 0: sprintf(buffer,"unknown"); break;
333             case 1: sprintf(buffer,"not raining"); break;
334             case 2: sprintf(buffer,"recently raining"); break;
335             case 3: sprintf(buffer,"raining"); break;
336             default: sprintf(buffer,"unknown"); break;
337         }
338         fprintf(fp,"%u (%s)\n",rainCond,buffer);
339         fprintf(fp,"%f\n",jd);
340         // SKYCOND CLOUD-SENSOR
341         fprintf(fp,"SKYCOND\n");
342         fprintf(fp,"CLOUD-SENSOR\n");
343         switch(skyCond) {
344             case 0: sprintf(buffer,"unknown"); break;
345             case 1: sprintf(buffer,"clear"); break;
346             case 2: sprintf(buffer,"cloudy"); break;
347             case 3: sprintf(buffer,"very cloudy"); break;
348             case 4: sprintf(buffer,"wet"); break;
349             default: sprintf(buffer,"unknown"); break;
350         }
351         fprintf(fp,"%u (%s)\n",skyCond,buffer);
352         fprintf(fp,"%f\n",jd);
353         // ROOFCLOSEREQUESTED CLOUD-SENSOR
354         fprintf(fp,"ROOFCLOSEREQUESTED\n");
355         fprintf(fp,"CLOUD-SENSOR\n");
356         switch(roofCloseRequested) {
357             case 0: sprintf(buffer,"normally"); break;
358             case 1: sprintf(buffer,"roof close"); break;
359             default: sprintf(buffer,"unknown"); break;

```

```

360
361     }
362     fprintf(fp, "%u (%s)\n", roofCloseRequested, buffer);
363     fprintf(fp, ".5f\n", jd);
364     // SKYMINUSAMBIENTTEMPERATURE CLOUD-SENSOR
365     fprintf(fp, "SKYMINUSAMBIENTTEMPERATURE\n");
366     fprintf(fp, "CLOUD-SENSOR\n");
367     fprintf(fp, ".1f\n", skyMinusAmbientTemperature);
368     fprintf(fp, ".5f\n", jd);
369     // AMBIENTTEMPERATURE CLOUD-SENSOR
370     fprintf(fp, "AMBIENTTEMPERATURE\n");
371     fprintf(fp, "CLOUD-SENSOR\n");
372     fprintf(fp, ".1f\n", ambientTemperature);
373     fprintf(fp, ".5f\n", jd);
374     // WINDSPEED CLOUD-SENSOR
375     fprintf(fp, "WINDSPEED\n");
376     fprintf(fp, "CLOUD-SENSOR\n");
377     fprintf(fp, ".1f\n", windSpeed);
378     fprintf(fp, ".5f\n", jd);
379     // WETSENSOR CLOUD-SENSOR
380     fprintf(fp, "WETSENSOR\n");
381     fprintf(fp, "CLOUD-SENSOR\n");
382     fprintf(fp, "%c\n", wetSensor);
383     fprintf(fp, ".5f\n", jd);
384     // RAINSENSOR CLOUD-SENSOR
385     fprintf(fp, "RAINSENSOR\n");
386     fprintf(fp, "CLOUD-SENSOR\n");
387     fprintf(fp, "%c\n", rainSensor);
388     fprintf(fp, ".5f\n", jd);
389     // RELATIVEHUMIDITYPERCENTAGE CLOUD-SENSOR
390     fprintf(fp, "RELATIVEHUMIDITYPERCENTAGE\n");
391     fprintf(fp, "CLOUD-SENSOR\n");
392     fprintf(fp, "%d\n", relativeHumidityPercentage);
393     fprintf(fp, ".5f\n", jd);
394     // DEWPOINTTEMPERATURE CLOUD-SENSOR
395     fprintf(fp, "DEWPOINTTEMPERATURE\n");
396     fprintf(fp, "CLOUD-SENSOR\n");
397     fprintf(fp, ".1f\n", dewPointTemperature);
398     fprintf(fp, ".5f\n", jd);
399     // CASETEMPERATURE CLOUD-SENSOR
400     fprintf(fp, "CASETEMPERATURE\n");
401     fprintf(fp, "CLOUD-SENSOR\n");
402     fprintf(fp, ".1f\n", caseTemperature);
403     fprintf(fp, ".5f\n", jd);
404     // RAINHEATERPERCENTAGE CLOUD-SENSOR
405     fprintf(fp, "RAINHEATERPERCENTAGE\n");
406     fprintf(fp, "CLOUD-SENSOR\n");
407     fprintf(fp, "%d\n", rainHeaterPercentage);
408     fprintf(fp, ".5f\n", jd);
409     // BLACKBODYTEMPERATURE CLOUD-SENSOR
410     fprintf(fp, "BLACKBODYTEMPERATURE\n");
411     fprintf(fp, "CLOUD-SENSOR\n");
412     fprintf(fp, ".1f\n", blackBodyTemperature);
413     fprintf(fp, ".5f\n", jd);
414     // RAINHEATERSTATE CLOUD-SENSOR
415     fprintf(fp, "RAINHEATERSTATE\n");
416     fprintf(fp, "CLOUD-SENSOR\n");
417     fprintf(fp, "%d\n", rainHeaterState);
418     fprintf(fp, ".5f\n", jd);
419     // POWERVOLTAGE CLOUD-SENSOR
420     fprintf(fp, "POWERVOLTAGE\n");
421     fprintf(fp, "CLOUD-SENSOR\n");
422     fprintf(fp, ".1f\n", powerVoltage);
423     fprintf(fp, ".5f\n", jd);
424     // ANEMOMETERTEMPERATUREDIFF CLOUD-SENSOR
425     fprintf(fp, "ANEMOMETERTEMPERATUREDIFF\n");
426     fprintf(fp, "CLOUD-SENSOR\n");
427     fprintf(fp, ".1f\n", anemometerTemeratureDiff);
428     fprintf(fp, ".5f\n", jd);
429     // WETNESSDROP CLOUD-SENSOR
430     fprintf(fp, "WETNESSDROP\n");
431     fprintf(fp, "CLOUD-SENSOR\n");
432     fprintf(fp, "%d\n", wetnessDrop);
433     fprintf(fp, ".5f\n", jd);
434     // WETNESSAVG CLOUD-SENSOR
435     fprintf(fp, "WETNESSAVG\n");
436     fprintf(fp, "CLOUD-SENSOR\n");
437     fprintf(fp, "%d\n", wetnessAvg);
438     fprintf(fp, ".5f\n", jd);
439     // WETNESSDRY CLOUD-SENSOR
440     fprintf(fp, "WETNESSDRY\n");
441     fprintf(fp, "CLOUD-SENSOR\n");
442     fprintf(fp, "%d\n", wetnessDry);
443     fprintf(fp, ".5f\n", jd);

```

```

443 // RAINHEATERPWM CLOUD-SENSOR
444 fprintf(fp, "RAINHEATERPWM\n");
445 fprintf(fp, "CLOUD-SENSOR\n");
446 fprintf(fp, "%d\n", rainHeaterPWM);
447 fprintf(fp, ".5f\n", jd);
448 // ANEMOMETERHEATERPWM CLOUD-SENSOR
449 fprintf(fp, "ANEMOMETERHEATERPWM\n");
450 fprintf(fp, "CLOUD-SENSOR\n");
451 fprintf(fp, "%d\n", anemometerHeaterPWM);
452 fprintf(fp, ".5f\n", jd);
453 // THERMOPILEADC CLOUD-SENSOR
454 fprintf(fp, "THERMOPILEADC\n");
455 fprintf(fp, "CLOUD-SENSOR\n");
456 fprintf(fp, "%u\n", thermopileADC);
457 fprintf(fp, ".5f\n", jd);
458 // THERMISTORADC CLOUD-SENSOR
459 fprintf(fp, "THERMISTORADC\n");
460 fprintf(fp, "CLOUD-SENSOR\n");
461 fprintf(fp, "%u\n", thermistorADC);
462 fprintf(fp, ".5f\n", jd);
463 // POWERADC CLOUD-SENSOR
464 fprintf(fp, "POWERADC\n");
465 fprintf(fp, "CLOUD-SENSOR\n");
466 fprintf(fp, "%u\n", powerADC);
467 fprintf(fp, ".5f\n", jd);
468 // BLOCKADC CLOUD-SENSOR
469 fprintf(fp, "BLOCKADC\n");
470 fprintf(fp, "CLOUD-SENSOR\n");
471 fprintf(fp, "%u\n", blockADC);
472 fprintf(fp, ".5f\n", jd);
473 // ANEMOMETERTHERMISTORADC CLOUD-SENSOR
474 fprintf(fp, "ANEMOMETERTHERMISTORADC\n");
475 fprintf(fp, "CLOUD-SENSOR\n");
476 fprintf(fp, "%u\n", anemometerThermistorADC);
477 fprintf(fp, ".5f\n", jd);
478 // DAVISVANEADC CLOUD-SENSOR
479 fprintf(fp, "DAVISVANEADC\n");
480 fprintf(fp, "CLOUD-SENSOR\n");
481 fprintf(fp, "%u\n", davisVaneADC);
482 fprintf(fp, ".5f\n", jd);
483 // DKMPH CLOUD-SENSOR
484 fprintf(fp, "DKMPH\n");
485 fprintf(fp, "CLOUD-SENSOR\n");
486 fprintf(fp, ".if\n", dkMPH);
487 fprintf(fp, ".5f\n", jd);
488 // EXTANEMOMETERTDIRECTION CLOUD-SENSOR
489 fprintf(fp, "EXTANEMOMETERTDIRECTION\n");
490 fprintf(fp, "CLOUD-SENSOR\n");
491 fprintf(fp, "%d\n", extAnemometerDirection);
492 fprintf(fp, ".5f\n", jd);
493 // RAWWETNESSOSC CLOUD-SENSOR
494 fprintf(fp, "RAWWETNESSOSC\n");
495 fprintf(fp, "CLOUD-SENSOR\n");
496 fprintf(fp, "%d\n", rawWetnessOsc);
497 fprintf(fp, ".5f\n", jd);
498 // DAYCOND CLOUD-SENSOR
499 fprintf(fp, "DAYCOND\n");
500 fprintf(fp, "CLOUD-SENSOR\n");
501 fprintf(fp, "%u\n", dayCond);
502 fprintf(fp, ".5f\n", jd);
503 // DAYLIGHTADC CLOUD-SENSOR
504 fprintf(fp, "DAYLIGHTADC\n");
505 fprintf(fp, "CLOUD-SENSOR\n");
506 fprintf(fp, "%u\n", daylightADC);
507 fprintf(fp, ".5f\n", jd);
508 // HEADER CLOUD-SENSOR
509 fprintf(fp, "HEADER\n");
510 fprintf(fp, "CLOUD-SENSOR\n");
511 fprintf(fp, "D E C W R i c SKY AMB WIND w r HUM DEW CASE HEA BLKT H ←
      PWR WNDTD WDRDP WAVG WDRY RHT AHT ASKY ACSE APSV ABLK AWND AVNE DKMPH←
      VNE RWOSC PH CN\n");
512 fprintf(fp, ".5f\n", jd);
513 // RAWDATA CLOUD-SENSOR
514 fprintf(fp, "RAWDATA\n");
515 fprintf(fp, "CLOUD-SENSOR\n");
516 fprintf(fp, "D %u %u %u %u %u %u %6.1f %5.1f %5.1f %c %c %3d %5.1f %5.1f ←
      %3d %5.1f %1d %4.1f %5.1f %5d %5d %5d %3d %3d %4u %4u %4u %4u %5u ←
      %4.1f %5d %3d %3u %2u\n",
      humidstatTempCode, cloudCond,
      windCond, rainCond, skyCond, roofCloseRequested,
      skyMinusAmbientTemperature, ambientTemperature, windSpeed,
      wetSensor, rainSensor, relativeHumidityPercentage,
      dewPointTemperature, caseTemperature, rainHeaterPercentage,

```

```

522     blackBodyTemperature, rainHeaterState, powerVoltage,
523     anemometerTemeratureDiff, wetnessDrop, wetnessAvg,
524     wetnessDry, rainHeaterPWM, anemometerHeaterPWM,
525     thermopileADC, thermistorADC, powerADC, blockADC,
526     anemometerThermistorADC, davisVaneADC, dkMPH,
527     extAnemometerDirection, rawWetnessOsc, dayCond,
528     daylightADC);
529     fprintf(fp,"%f\n",jd);
530 // CLOSE OUTPUT FILE
531     fclose(fp);
532     retval=1;
533 } else {
534     retval=0;
535 }
536 } else {
537     retval=0;
538 }
539 } else {
540     retval=0;
541 }
542 return retval;
543 }
544
545 int main(int argc, char **argv) {
546     double jd;
547     int count;
548
549 // ARGUMENTS CHECK
550 if(argc!=3) {
551     printf("%s device outfile\n",argv[0]);
552     return 0;
553 }
554 // MAIN LOOP
555 while(1) {
556     port_open(argv[1]);
557     count=0;
558     while(count<5 && fd!=-1) {
559         jd=read_time();
560         if(fd==-1) break;
561         count=read_data(jd,argv[2])?0:count+1;
562         if(fd==-1) break;
563         usleep(100000);
564     }
565     port_close();
566     sleep(1);
567 }
568 // NEVER RETURN!!!
569 return 0;
570 }
```

Código 44: Contenido del fichero cloudsensor.c

12.4.8. sunpos.py

```

1 #!/usr/bin/env python
2
3 import datetime
4 import ephem
5
6 o = ephem.Observer()
7
8 o.lat="42:03:05"
9 o.long="0:43:46"
10 o.elevation=1570.0
11 o.date=datetime.datetime.utcnow()
12
13 sun = ephem.Sun(o)
14
15 print "SOLARPOS"
16 print "SUNPOSITION"
17 print sun.alt*180/ephem.pi
18 print ephem.julian_date(o.date)
19 print "LATITUDE"
20 print o.lat
21 print "LONGITUDE"
22 print o.long
```

```

23 print "ELEVATION"
24 print o.elevation
25 print "DATETIME"
26 print o.date
27 print "AZIMUTH"
28 print sun.az*180/ephem.pi
29 print "ALTITUDE"
30 print sun.alt*180/ephem.pi

```

Código 45: Contenido del fichero sunpos.py

12.4.9. snmpd.conf

```

1 ######
2 #
3 # snmpd.conf:
4 #
5 #####
6
7 #      sec.name   source           community
8 com2sec local    localhost        public
9 com2sec network  default         public
10
11 #      groupName      securityModel securityName
12 group RWGroup v1      local
13 group RWGroup v2c     local
14 group RWGroup usm    local
15 group ROGroup v1     network
16 group ROGroup v2c    network
17 group ROGroup usm   network
18
19 # Make at least snmpwalk -v 1 localhost -c public system fast again.
20 #          name       incl/excl    subtree        mask(optional)
21 view all   included   .1           80
22
23 #      group      context sec.model sec.level prefix read  write  notif
24 access ROGroup ""     any      noauth   exact  all    none   none
25 access RWGroup ""     any      noauth   exact  all    all    none
26
27 # System contact information
28
29 syslocation Linux (Fedora 14), Personal Laptop.
30 syscontact Josep Sanz <josep.sanz@saltos.net>
31
32 # Logging
33
34 dontLogTCPWrappersConnects yes
35
36 # Further Information
37
38 extend .1.3.6.1.4.1.2021.50.1 davis /bin/cat /tmp/davis.txt
39 extend .1.3.6.1.4.1.2021.50.2 raindetect /bin/cat /tmp/raindetect.txt
40 extend .1.3.6.1.4.1.2021.50.3 previstorm /bin/cat /tmp/previstorm.txt
41 extend .1.3.6.1.4.1.2021.50.4 emsmc /bin/cat /tmp/emsmc.txt
42 extend .1.3.6.1.4.1.2021.50.5 date /bin/date
43 extend .1.3.6.1.4.1.2021.50.6 sunpos /usr/bin/python /home/optjo/gitrepos/tjo/rebei/sunpos.py
44 extend .1.3.6.1.4.1.2021.50.7 cloudsensor /bin/cat /tmp/cloudsensor.txt
45 extend .1.3.6.1.4.1.2021.50.8 vaisala /bin/cat /tmp/vaisala.txt

```

Código 46: Contenido del fichero snmpd.conf

12.5. Programación del diseño elegido

12.5.1. orocs

```

1#!/usr/bin/php -q
2<?php
3// TO FIND ALL REQUIRED FILES

```

```

4  chdir(dirname($_SERVER["SCRIPT_NAME"]));
5  // INCLUDES OF ALL SALTOS FILES
6  include("php/xml2array.php");
7  include("php/array2xml.php");
8  include("php/saltos.php");
9  // INCLUDE OF ALL OPENROCS FILES
10 include("php/functions.php");
11 include("php/process.php");
12 include("php/define.php");
13 include("php/childs.php");
14 include("php/checks.php");
15 include("php/stack.php");
16 include("php/pipes.php");
17 include("php/comm.php");
18 include("php/gc.php");
19 // CONTINUE
20 program_error_handler();
21 if(!ini_get("date.timezone")) ini_set("date.timezone","Europe/Madrid");
22 set_time_limit(0);
23 declare(ticks=100);
24 garbage_collector_init();
25 $file=xml_real_file(__XML_CONFIG__);
26 if(!file_exists($file)) show_php_error(array("phperror"=>"Config file not found"));
27 $config=xml2array($file);
28 $config=check_config($config,xml_real_file(__XML_CONFIG__));
29 eval_iniset(getNode(__INI_SET__));
30 if(getNode(__PUTENV_PATH__)) list($config[__PUTENV__][__PATH__],$error)=str_replace_with_vars←
    (getNode(__PUTENV_PATH__),config_array_vars());
31 eval_putenv(getNode(__PUTENV__));
32 include("php/args.php");
33 ?>

```

Código 47: Contenido del fichero orocs de OpenROCS v2.0

12.6. Funciones heredadas del proyecto SaltOS

12.6.1. array2xml.php

Este fichero aporta el código para poder convertir estructuras de datos de tipo *Array* a estructuras de datos de tipo *XML*.

```
function __escribir_nodos_check_node_name($name)
```

Esta función es un helper que permite validar que el nombre del nodo que se va a generar es válido según las especificaciones del formato *XML*.

```
function escribir_nodos(&$array,$level=null)
```

Esta función es un helper que realiza la tarea de generar los nodos completos del *XML* a partir del *Array* de entrada. El parámetro *level* indica el nivel de profundidad del nodo (usado para la indentación del código generado). Esta función usa recursividad para procesar los nodos que contienen hijos.

```
function array2xml($array)
```

Esta función es la función principal del conversor de *Array* a *XML*. retorna un string con el código XML resultante de la conversión.

12.6.2. saltos.php

Este fichero aporta el código para la gestión de errores, gestión de semáforos y otras funcionalidades como el control de números pseudo-aleatorios, gestión de ficheros, gestión de logs, gestión de memoria, gestión de cadenas, ...

```
function capture_next_error()
```

Esta función activa el control de errores para interceptar un error sin perder el control del flujo de ejecución. Esta función tiene un control de anidación, es decir, se puede llamar n veces y cada vez, guarda los errores en una posición del stack diferente. Esto es especialmente útil cuando se llama a funciones que internamente también usan este control de errores.

```
function get_clear_error()
```

Esta función recupera el último error (si ha existido) y desactiva el control de errores activado por la llamada a *capture_next_error*.

```
function do_message_error($array,$format)
```

Esta función genera un report humano a partir de la información de un error pasado por parámetro.

```
function __debug_backtrace_helper(&$item,$key)
```

Esta función es un helper de la función *do_message_error* y generará la parte del report que indica la traza de la ejecución hasta el momento del error.

```
function __do_message_error_helper(&$item,$key,$dict)
```

Esta función es un helper de la función *do_message_error* y ayuda a esta función para generar el report final que retornará la función a la que ayuda.

```
function show_php_error($array=null)
```

Esta función lanza un error controlado. Para ello, se sirve del parámetro *array* que aporta toda la información del error y sirve para generar un report personalizado dependiendo del tipo de error. Esta función puede ser llamada por el usuario y por los handlers de control de errores.

```
function __error_handler($type,$message,$file,$line)
```

Esta función es la que se programa mediante la función *set_error_handler* y sirve para interceptar los errores que se produzcan en la ejecución de OpenROCS.

```
function __exception_handler($e)
```

Esta función es la que se programa mediante la función *set_exception_handler* y sirve para interceptar las excepciones que se produzcan en la ejecución de OpenROCS.

```
function __shutdown_handler()
```

Esta función es la que se programa mediante la función *register_shutdown_function* y se programa para que cuando finalice la ejecución del script, se ejecute esta función y libere los semáforos que estén en uso en ese instante.

```
function program_error_handler()
```

Esta función configura el entorno de PHP para que use el control de errores proporcionado por SaltOS.

```
function init_random()
```

Esta función inicializa el sistema de generación de números pseudo-aleatorios.

```
function get_unique_id_md5()
```

Esta función genera un hash en formato MD5 único. Es usado para definir las cadenas de identificación entre procesos, de entre otros.

```
function get_temp_file($hash=')
```

Esta función proporciona un nombre de fichero temporal único. Empleado en varias partes de OpenROCS para definir mecanismos de comunicación entre procesos y OpenROCS.

```
function current_datetime($offset=0)
```

Retorna la marca de tiempo en formato "AAAA-MM-DD HH:MM:SS"

```
function current_date($offset=0)
```

Retorna la marca de tiempo en formato "AAAA-MM-DD"

```
function current_time($offset=0)
```

Retorna la marca de tiempo en formato "HH:MM:SS"

```
function current_datetime_decimals($offset=0)
```

Retorna la marca de tiempo en formato "AAAA-MM-DD HH:MM:SS.DDDD"

```
function current_decimals($offset=0)
```

Retorna la marca de tiempo en formato "DDDD"

```
function __addlog_helper($a)
```

Esta función es un helper para la función *addlog*

```
function checklog($hash,$file="")
```

Esta función valida que el error con el *hash* pasado por argumento no exista en el fichero *file*.

```
function addlog($msg,$file)
```

Esta función añade el mensaje *msg* en el fichero *file*. También se encarga de mantener el sistema de rotación de logs (basado en el número de líneas) y de formatear el mensaje para que sea de fácil interpretación mediante la adición de una marca de tiempo y la alineación del mensaje para que se pueda visualizar correctamente en el fichero.

```
function getNode($path,$array=null)
```

Esta función retorna el nodo solicitado por el parámetro *path* de la estructura de tipo *array*. Si no se encuentra el nodo solicitado, retorna *null*.

```
function __getNode_helper($path,$array)
```

Esta función es un helper de la función *getNode* y permite moverse por la estructura *array* teniendo en cuenta que internamente, existen nodos que tienen valores y atributos.

```
function __getValue_helper($array)
```

Esta función es un helper de la función *getNode* y retorna el valor del nodo teniendo en cuenta que internamente, existen nodos que tienen valores y atributos.

```
function eval_iniset($array)
```

Esta función establece los parámetros de php mediante llamadas a *ini_set* de los parámetros pasados por argumento en la variable *array*.

function eval_putenv(\$array)

Esta función establece los parámetros del entorno mediante llamadas a putenv de los parámetros pasados por argumento en la variable *array*.

function encode_bad_chars(\$cad)

Esta función normaliza un string usando los siguientes caracteres ASCII:

- Letras en minúsculas y mayúsculas.
- Números del 0 al 9.
- Espacios.

Además, elimina caracteres innecesarios (dos espacios juntos los convierte en uno) y quita los espacios del principio y final de la cadena.

function sprintf(\$array)

Esta función es equivalente a llamar a print_r(\$array,true), y existe para dar compatibilidad a versiones de PHP que no disponen de esta función.

function semaphore_acquire(\$file,\$timeout=100000)

Esta función sirve para solicitar el bloqueo de un semáforo. El semáforo se identifica por el fichero *file*. Retorna un booleano indicando si el bloqueo se ha podido conseguir o no (esto viene dado por el parámetro *timeout*).

function semaphore_release(\$file)

Esta función sirve para solicitar el desbloqueo de un semáforo.

function semaphore_shutdown()

Esta función sirve para solicitar el bloqueo de todos los semáforos existentes.

function __semaphore_helper(\$fn,\$file,\$timeout)

Esta función es un helper de las funciones de control de semáforos. Realmente, las funciones anteriores, llaman a esta función que es la que implementa toda la lógica de control de semáforos y mediante el parámetro *fn* identifica la funcionalidad deseada en la llamada. Es una manera de programar toda la funcionalidad en una única función y mediante inmersión, separar en funciones individuales cada una de las funcionalidades que esta ofrece.

```
function memory_get_free()
```

Retorna la memoria libre del interprete de PHP.

```
function usleep_protected($usec)
```

Esta función emula la función *usleep*, pero usando *sockets* en lugar de hacer una llamada a *usleep*. La función *usleep* bloquea la ejecución y no se despierta hasta que ha transcurrido el tiempo solicitado, pero este comportamiento tiene un problema grave en OpenROCS: ¿que pasa si este proceso debe procesar una señal (signal)?. Mediante el uso de esta función, se garantiza que el proceso dormirá el tiempo solicitado, pero en caso de que una señal llegue al proceso que está durmiendo, esta función se desbloqueará, permitiendo que continúe la ejecución del código y atienda la señal, tal como se espera. Para controlar si se ha dormido el tiempo solicitado o no, esta función retorna el valor de tiempo que realmente ha dormido.

```
function chmod_protected($file,$mode)
```

Esta función ejecuta la función *chmod* en modo protegido, controlando errores que puedan suceder. Retorna un string indicando el error en caso de que haya sucedido.

```
function touch_protected($file)
```

Esta función ejecuta la función *touch* en modo protegido, controlando errores que puedan suceder. Retorna un string indicando el error en caso de que haya sucedido.

12.6.3. [xml2array.php](#)

Este fichero, proporciona las funciones que permiten a OpenROCS convertir ficheros en formato *XML* a formato *array* para el uso interno de la aplicación.

```
function leer_nodos(&$data,$file="")
```

Esta función es un helper de la función *xml2array* y generá una estructura de árbol a partir de los datos que le llegan por el parámetro *data*.

```
function set_array(&$array,$name,$value)
```

Esta función es un setter que permite añadir en el *array* el valor de *value* con nombre *name*. Si *name* ya existe, se genera una nueva entrada en el *array* añadiendo un sufijo del tipo *#n*, donde *n* es un valor numérico que es único para cada entrada.

```
function unset_array(&$array,$name)
```

Esta función elimina entradas del *array* cuyo nombre coincide con *name*. Si existen entradas con sufijo del tipo *#n*, también serán borradas.

```
function limpiar_key($key)
```

Esta función retorna el nombre de la entrara *key* sin el sufijo del tipo *#n*, si existe.

```
function xml2array($file)
```

Esta función convierte el fichero *file* en un *array*, es decir, en una estructura de árbol.

```
function xml2struct($xml,$file=")
```

Esta función es un helper de la función *xml2array* y genera una lista de los nodos del *xml* pero sin usar una estructura de árbol. Esta lista es la que posteriormente se pasa a la función *leer_nodos* para convertirla en una estructura de árbol. Puede recibir un parámetro opcional llamado *file* que permite en caso de error en el procesado del *xml* poder indicar en que fichero se ha encontrado el error.

```
function eval_bool($arg)
```

Esta función permite evaluar valores booleanos de diferentes tipos:

- Los valores TRUE, ON, YES y 1 se evalúan como cierto.
- Los valores FALSE, OFF, NO y 0 se evalúan como falso.
- El resto de valores lanzan un error indicando que el tipo de booleano es desconocido.

```
function xml_error($error,$source="",$count="",$file="")
```

Esta función sirve para canalizar todos los errores que se generan al procesar ficheros XMLs.

12.7. Funciones y servicios de OpenROCS

12.7.1. args.php

Este fichero implementa el workflow del cliente (comando *orocs*). Es el fichero encargado de cargar los ficheros de acciones y/o ejecutar llamadas a funciones para generar la salida esperada. Todas las ejecuciones del cliente *orocs* pasan por este fichero.

12.7.2. broadcast.php

Este fichero contiene el código que implementa el servicio de broadcast. Para ello, usa el siguiente workflow:

- Se programan las señales (SIGTERM, SIGINT, SIGCHLD y SIGUSR1).
- Lee el hash que le envía el proceso padre. Este hash servirá durante toda la vida del proceso para comunicarse con el resto de procesos.

- Inicializa el garbage collector.
- Pone el estado del proceso a *stop*.
- define y programa otra función *shutdown_handler*.
- Crea un servidor UDP/IP.
- En caso de error, lo notifica al proceso padre y finaliza la ejecución.
- Si todo va bien, notifica al padre que se ha activado el servicio.
- Se queda esperando a que alguien ponga el proceso en estado *start*.
- Entra en un bucle infinito hasta que se destruya el socket para realizar las siguientes tareas:
 - Atender las peticiones del servidor UDP/IP.
 - Enviar paquetes a la dirección de broadcast para intentar descubrir nuevas instancias de OpenROCS.
 - Las instancias que desaparecen, son borradas de la lista de instancias descubiertas.
 - Sincronizar los datos desde las instancias de OpenROCS descubiertos hacia el OpenROCS local.

function shutdown_handler()

Esta función, es específica de este servicio y tiene las siguientes tareas asociadas:

- Cerrar todos los sockets activos del proceso.
- Cerrar el garbage collector.

12.7.3. checks.php

Este fichero define las funciones que OpenROCS emplea para hacer chequeos del sistema.

function check_system()

Esta función realiza un test de la gestión de procesos y de la gestión de semáforos. Útil para comprobar que se mantiene la integridad del sistema.

function check_config(\$config,\$file)

Esta función realiza un test para validar que el fichero de configuración general *config.xml* es correcto y que todos los nodos y valores contienen datos válidos y coherentes.

function check_monitor(\$monitor,\$file)

Esta función realiza un test para validar que el fichero de configuración del monitor *monitor.xml* es correcto y que todos los nodos y valores contienen datos válidos y coherentes.

```
function check_scheduler($scheduler,$file)
```

Esta función realiza un test para validar que el fichero de configuración del scheduler *scheduler.xml* es correcto y que todos los nodos y valores contienen datos válidos y coherentes.

12.7.4. childs.php

Este fichero contiene el código necesario para la gestión de procesos.

```
class child_helper
```

La clase *child_helper*, tal como su nombre indica, es un helper para la gestión de procesos. Básicamente se trata de una clase que contiene 4 propiedades (pid, stdout, stderr y timestamp) y permite gestionar de manera fácil el estado de un proceso, el tiempo empleado por el proceso, los canales stdout y stderr, enviar señales en caso de necesidad, ...

```
class child_helper::function __construct($cmd)
```

El constructor de esta clase es el encargado de ejecutar el comando mediante la llamada a la función *child_exec* y guardar en las propiedades los datos devueltos por esta función.

```
class child_helper::function __destruct()
```

El destructor de esta clase es el encargado de limpiar la memoria usada por este objeto. Básicamente realiza una limpieza de los canales stdout y stderr.

```
class child_helper::function get_pid()
```

Este método de la clase retorna el PID del proceso.

```
class child_helper::function get_stdout()
```

Este método de la clase retorna los datos del canal stdout del proceso.

```
class child_helper::function get_stderr()
```

Este método de la clase retorna los datos del canal stderr del proceso.

```
class child_helper::function get_timestamp()
```

Este método de la clase retorna la marca de tiempo del instante en que se lanzó la ejecución del proceso.

```
class child_helper::function set_timestamp($timestamp)
```

Este método de la clase establece la marca de tiempo del proceso al valor pasado por parámetro.

```
class child_helper::function exists()
```

Este método de la clase retorna si el proceso existe o no.

```
class child_helper::function elapsed_time($timestamp=")
```

Este método de la clase retorna el tiempo transcurrido desde que se lanzó la ejecución del proceso.

```
class child_helper::function wait_while_exists()
```

Este método de la clase realiza una espera hasta que el proceso dejé de existir.

```
class child_helper::function kill($signal,$recursive=false)
```

Este método de la clase envía la señal *signal* al proceso. El parámetro *recursive* indica al método que debe enviar la señal tanto al proceso como a los hijos del proceso. Es útil cuando se ejecutan comandos que lanzan otros procesos.

```
class child_helper::function sigterm_sigkill()
```

Este método de la clase envía la señal *SIGTERM* al proceso. En caso de que este no responda a la señal, se le envía la señal *SIGKILL*.

```
function child_kill($pid,$signal,$recursive=false)
```

Ídem que el método *kill* de la clase *child_helper*.

```
function __child_kill_recursive($pid)
```

Esta función es un helper de la función *child_kill* y tiene la funcionalidad de retornar los PIDs de los procesos hijos. Como ejemplo, si se llama a esta función solicitando los PIDs de los procesos hijos del proceso con PID=1, esta función deberá retornar los PIDs de todos los procesos que existen en el sistema, puesto el el proceso con PID=1 es el proceso *INIT*.

```
function child_wait($pid)
```

Ídem que el método *wait_while_exists* de la clase *child_helper*.

```
function child_sigterm_sigkill($pid)
```

Ídem que el método *sigterm_sigkill* de la clase *child_helper*.

```
function child_exec($cmd)
```

Esta función lanza la ejecución de un proceso y retorna una estructura con el pid, identificador del canal del stdout y stderr, y timestamp.

```
function child_exists($pid)
```

Ídem que el método *exists* de la clase *child_helper*.

```
function child2hash($host,$port,$name,$pid,$node,$alias,$pipe)
```

Esta función genera un cadena en base64 con la información del proceso (*host*, *port*, *name*, *pid*, *node*, *alias* y *pipe*).

```
function hash2array($hash)
```

Esta función realiza el paso inverso a la anterior. A partir del *hash* (que en realidad es una cadena en base64), retorna una estructura con la información original.

```
function child_host($hash)
```

Esta función retorna el *host* contenido en el *hash*.

```
function child_port($hash)
```

Esta función retorna el *port* contenido en el *hash*.

```
function child_name($hash)
```

Esta función retorna el *name* contenido en el *hash*.

```
function child_pid($hash)
```

Esta función retorna el *pid* contenido en el *hash*.

```
function child_node($hash)
```

Esta función retorna el *node* contenido en el *hash*.

```
function child_alias($hash)
```

Esta función retorna el *alias* contenido en el *hash*.

```
function child_pipe($hash)
```

Esta función retorna el *pipe* contenido en el *hash*.

```
function child_string($hash)
```

Esta función retorna una cadena usada para mostrarla al usuario que identifica el proceso con sus datos más característicos.

```
function child_key_val($hash)
```

Esta función retorna dos datos (clave y valor) a partir de una cadena de entrada con el formato CLAVE=VALOR.

```
function child_send($hash,$cmd,$arg="")
```

Esta función envía señales entre procesos. El destinatario se identifica con el *hash* y se enviarán dos datos: *cmd* como mensaje y permite adjuntar argumentos en el campo *arg*. Esta función es síncrona, es decir, que retornará cuando el proceso destinatario haya procesado el mensaje y haya enviado una respuesta. La respuesta es el valor de retorno de la función.

```
function child_read($hash)
```

Esta función realiza la lectura de la pipe del proceso contenido en *hash*.

```
function child_response($hash,$data="")
```

Esta función realiza la escritura del contenido del parámetro *data* en la pipe del proceso contenido en *hash*.

12.7.5. comm.php

Este fichero proporciona la funcionalidad de comunicación con el servidor de OpenROCS.

```
function comm($cmd,$host_extra="",$port_extra="")
```

Esta función envía un comando al servidor de OpenROCS y espera hasta que este recibe la respuesta completa. Los comandos que se pueden usar con esta función son todos los comandos soportados por el servidor.

Internamente, se usa un control de CRC32 para validar que el mensaje se ha recibido de forma correcta. Para ello, esta función añade siempre al final de todas las peticiones, el comando *crc32*, tal como está documentado en la implementación del *server.php*.

Esta función también implementa un sistema de reintentos, cosa que permite que dado un error en la comunicación, la función tenga la inteligencia de volver a intentar obtener el resultado esperado. El valor retornado por el servidor es el valor de retorno de esta función (que es una cadena de texto).

En caso de no recibir respuesta del servidor (porque este detenido el servicio), esta función retornará *false*.

También permite definir 2 parámetros adicionales (*host_extra* y *port_extra*), usado cuando se quiere enviar comandos a otras instancias de OpenROCS. Si estos parámetros se omiten, se usan los valores por defecto de la instancia de OpenROCS que lo ejecuta.

```
function comm_get_array($cmd,$host="",$port="")
```

Esta función es idéntica a la función *comm* (de hecho, es una inmersión de esta función) pero se diferencia por la manera en que retorna el resultado. Mientras que la función *comm* retorna un string, esta función retorna un array con el resultado en forma de lista sin la cabecera. Es útil cuando se desea conocer todos los STACKS del servidor o todas las variables de un STACK.

Al igual que la función *comm*, también se pueden pasar de forma opcional los parámetros *host* y *port*.

```
function comm_and_wait($cmd,$host="",$port="")
```

Esta función es idéntica a la función *comm* (de hecho, es una inmersión de esta función) pero se diferencia por el hecho de que si detecta que el comando a enviar al servidor es un comando de tipo *start* o *stop*, realizará un bloqueo hasta que el estado del proceso al cual se ha enviado ese comando sea el mismo que el deseado por el comando.

Al igual que la función *comm*, también se pueden pasar de forma opcional los parámetros *host* y *port*.

```
function comm_human($cmd,$host="",$port="")
```

Esta función es idéntica a la función *comm* (de hecho, es una inmersión de esta función) pero se diferencia por que únicamente, en caso de no recibir respuesta del servidor (es decir, si el valor retornado por *comm* es *false*, entonces, esta función lo reemplazará por un mensaje indicando que el servidor no esta funcionando).

Al igual que la función *comm*, también se pueden pasar de forma opcional los parámetros *host* y *port*.

12.7.6. define.php

Este fichero sirve a OpenROCS como fichero de configuración interno. Este fichero contiene todos los *defines* usados para definir:

- El nombre de las acciones que soporta OpenROCS.
- Los ficheros que implementan las acciones.

- Caracteres que se usan frecuentemente y con significados especiales.
- Cadenas usadas en el interfaz de usuario.
- Ficheros de log predeterminados.
- Variables internas.
- Señales que se usarán y directorios especiales.
- Semáforos para las funciones.
- Ficheros de configuración XML.
- Rutas para acceder a determinados nodos de información.
- Nombres de nodos para funcionalidades concretas.
- ...

```

1  <?php
2 // IMPORTANT INTERNALS STRINGS
3 define("__SHELL_CMD__","shell");
4 define("__HELP_CMD__","help");
5 define("__START_CMD__","start");
6 define("__RESTART_CMD__","restart");
7 define("__RELOAD_CMD__","reload");
8 define("__STOP_CMD__","stop");
9 define("__STATUS_CMD__","status");
10 define("__GET_CMD__","get");
11 define("__ADD_CMD__","add");
12 define("__CREATE_CMD__","create");
13 define("__UPDATE_CMD__","update");
14 define("__SET_CMD__","set");
15 define("__REMOVE_CMD__","remove");
16 define("__DELETE_CMD__","delete");
17 define("__DUMP_CMD__","dump");
18 define("__CHECK_CMD__","check");
19 define("__CRONTAB_CMD__","crontab");
20 define("__CRON_CMD__","cron");
21 define("__EXIT_CMD__","exit");
22 define("__QUIT_CMD__","quit");
23 define("__BYE_CMD__","bye");
24 define("__LOG_CMD__","log");
25 define("__HISTORY_CMD__","history");
26 define("__CHILD_ARG__","CHLD");
27 define("__CRC32_CMD__","CRC32");
28 // PHP FILES
29 define("__SHELL_PHP__","php/action/shell.php");
30 define("__HELP_PHP__","php/action/help.php");
31 define("__START_PHP__","php/action/start.php");
32 define("__START0_PHP__","php/action/start0.php");
33 define("__START1_PHP__","php/action/start1.php");
34 define("__START2_PHP__","php/action/start2.php");
35 define("__RESTART_PHP__","php/action/restart.php");
36 define("__RELOAD_PHP__","php/action/reload.php");
37 define("__STOP_PHP__","php/action/stop.php");
38 define("__STOP1_PHP__","php/action/stop1.php");
39 define("__STOP2_PHP__","php/action/stop2.php");
40 define("__CRONTAB_PHP__","php/action/crontab.php");
41 define("__SERVER_PHP__","php/server.php");
42 define("__MONITOR_PHP__","php/monitor.php");
43 define("__SCHEDULER_PHP__","php/scheduler.php");
44 define("__BROADCAST_PHP__","php/broadcast.php");
45 // SOME CHARACTERS
46 define("__EOL__","\n");
47 define("__NONE__","");
48 define("__SPACE__","");
49 define("__QUOTE__",'\"');
50 define("__EQUAL__","=");
51 define("__DOLLAR__",'$');
52 define("__UNDERSTAND__",___);
53 define("__NEGATION__",__!');
54 define("__XPATH__",__ "/");
55 define("__TWO_POINTS__",__:");
56 define("__ARROBA__",__@__);
57 // USER INTERFACE STRINGS

```

```

58 define("__SHELL_WELCOME__", implode(array_map("chr", array<→
      (32,32,95,95,.....,32,124,95,124,10)))); 
59 define("__SHELL_PROMPT__", "orocs#");
60 define("__SHELL_RUNNING__", "Shell is running");
61 define("__UNKNOWN_COMMAND__", "Unknown command");
62 define("__UNKNOWN_PARAMETER__", "Unknown parameter");
63 define("__UNKNOWN_STACK__", "Unknown stack");
64 define("__UNKNOWN_DATA__", "Unknown data");
65 define("__UNKNOWN_SERVICE__", "Unknown service");
66 define("__PERMISSION_DENIED__", "Permission denied to");
67 define("__SERVER_ALREADY_RUNNING__", "The server is already running");
68 define("__SERVER_NOT_RUNNING__", "The server is not running");
69 define("__SERVER_RUNNING__", "Server running");
70 define("__SERVER_START__", "Server up, listening on port");
71 define("__SERVER_ERROR__", "Server boot error");
72 define("__SERVER_STOPPING__", "Stopping the server");
73 define("__SERVER_STARTING__", "Starting the server");
74 define("__DATA_FOUND__", "data(s) found in");
75 define("__DATA_ADDED__", "data(s) added in");
76 define("__DATA_UPDATED__", "data(s) updated in");
77 define("__DATA_REMOVED__", "data(s) removed in");
78 define("__DATA_NOT_FOUND__", "Data not found");
79 define("__STACK_FOUND__", "stack(s) found");
80 define("__STACK_ADDED__", "stack(s) added");
81 define("__STACK_REMOVED__", "stack(s) removed");
82 define("__STACK_NOT_FOUND__", "Stack not found");
83 define("__CHILD_CREATED__", "Created new child");
84 define("__WATCHDOG_STARTED__", "Watchdog started in");
85 define("__WATCHDOG_STOPPED__", "Watchdog stopped in");
86 define("__SENDING_TERM__", "Sending term signal to");
87 define("__SENDING_KILL__", "Sending kill signal to");
88 define("__MONITOR_START__", "Monitor up");
89 define("__SCHEDULER_START__", "Scheduler up");
90 define("__NOT_FOUND__", "not found");
91 define("__NOT_NUMERIC__", "not numeric");
92 define("__GREATER_ZERO__", "must be greater than zero");
93 define("__REQUIRES__", "requires");
94 define("__OR__", "or");
95 define("__AND__", "and");
96 define("__MONITOR_NOT_FOUND__", "Monitor not found");
97 define("__TASK_NOT_FOUND__", "Task not found");
98 define("__NOT_DETECT__", "not detect");
99 define("__ONLY_ONE__", "can not be defined at the same time");
100 define("__COLLISION_ERROR__", "Detected collisions between variables");
101 define("__SERVICE__", "Service");
102 define("__IS__", "is");
103 define("__IS_GOING_TO__", "is going to");
104 define("__SCHEDULER_NOT_FOUND__", "Scheduler not found");
105 define("__ACTIONS_NOT_FOUND__", "Actions not found");
106 define("__BROADCAST_START__", "Broadcast up, listening on port");
107 define("__BROADCAST_ERROR__", "Broadcast boot error");
108 define("__VALUE_ERROR__", "value error");
109 define("__SENDING_START__", "Sending start to");
110 define("__SENDING_STOP__", "Sending stop to");
111 define("__SAVING_DATA__", "Saving data ...");
112 define("__RESTORING_DATA__", "Restoring data ...");
113 define("__ENTRIES_SAVED__", "entries saved");
114 define("__ENTRIES_RESTORED__", "entries restored");
115 // LOG FILES
116 define("__DEFAULT_LOG__", "log/orocs.log");
117 define("__ERROR_LOG__", "log/error.log");
118 define("__WARNING_LOG__", "log/warning.log");
119 define("__TRACE_LOG__", "log/trace.log");
120 define("__DEBUG_LOG__", "log/debug.log");
121 define("__USER_LOG__", "log/user.log");
122 // INTERNAL DEFINES
123 define("__HELPER_CMD__", implode(array_map("chr", array(120,121,122,122,121)))); 
124 define("__HELPER_MSG__", implode(array_map("chr", array(32,32,32,32,.....,124,95,124,10)))); 
125 define("__HOME__", "HOME");
126 define("__HOSTNAME__", "HOSTNAME");
127 define("__PATH__", "PATH");
128 define("__MAXLINES__", 1000);
129 define("__MAXSIZE__", 1024);
130 // SIGNALS
131 define("__SIGNAL_TERM__", SIGTERM);
132 define("__SIGNAL_INT__", SIGINT);
133 define("__SIGNAL_CHLD__", SIGCHLD);
134 define("__SIGNAL_USR1__", SIGUSR1);
135 define("__SIGNAL_KILL__", SIGKILL);
136 define("__PROC_DIR__", "/proc/");
137 define("__PROC_STAT__", "/stat");
138 // SEMAPHORES
139 define("__SEMAPHORE_COMM__", get_temp_file(md5("__SEMAPHORE_COMM__")));

```

```

140 // XML FILES
141 define("__XML_CONFIG__","xml/config.xml");
142 define("__XML_MONITOR__","xml/monitor.xml");
143 define("__XML_SCHEDULER__","xml/scheduler.xml");
144 define("__XML_VARIABLES__","xml/variables.xml");
145 define("__XML_ACTIONS__","xml/");
146 // XML PATHS OF CONFIG.XML
147 define("__SERVER_NODE__","server");
148 define("__SERVER_HOST__","server/host");
149 define("__SERVER_PORT__","server/port");
150 define("__SERVER_NAME__","server/name");
151 define("__SERVER_STACKS__","server/stacks");
152 define("__SERVER_PRIORITY__","server/priority");
153 define("__SERVER_PRIORITY_ENABLED__","server/priority/enabled");
154 define("__SERVER_PRIORITY_VALUE__","server/priority/value");
155 define("__BROADCAST_NODE__","broadcast");
156 define("__BROADCAST_ENABLED__","broadcast/enabled");
157 define("__BROADCAST_PORT__","broadcast/port");
158 define("__BROADCAST_DISCOVERY__","broadcast/discovery");
159 define("__BROADCAST_SYNCHRONIZE__","broadcast/synchronize");
160 define("__DEBUG_COMM__","debug/comm");
161 define("__DEBUG_PROCESS__","debug/process");
162 define("__DEBUG_SIGNAL__","debug/signal");
163 define("__DEBUG_TRACE__","debug/trace");
164 define("__DEBUG_MAXLINES__","debug/maxlines");
165 define("__DEBUG_PERCENT__","debug/percent");
166 define("__DEBUG_STATISTICS__","debug/statistics");
167 define("__SHELL_HISTORY__","shell/history");
168 define("__SHELL_MAXLINES__","shell/maxlines");
169 define("__TIMEOUT_COMM__","timeout/comm");
170 define("__TIMEOUT_CHILDREN__","timeout/child");
171 define("__TIMEOUT_PIPES__","timeout/pipes");
172 define("__TIMEOUT_WAIT__","timeout/wait");
173 define("__TIMEOUT_SERVER__","timeout/server");
174 define("__TIMEOUT_SEMAPHORE__","timeout/sema");
175 define("__POLLING_COMM__","polling/comm");
176 define("__POLLING_CHILDREN__","polling/child");
177 define("__POLLING_PIPES__","polling/pipes");
178 define("__POLLING_WAIT__","polling/wait");
179 define("__POLLING_SERVER__","polling/server");
180 define("__POLLING_MONITOR__","polling/mon");
181 define("__POLLING_SCHEDULER__","polling/sched");
182 define("__POLLING_BROADCAST__","polling/broadcast");
183 define("__RETRIES_COMM__","retries/comm");
184 define("__RETRIES_CHILDREN__","retries/child");
185 define("__RETRIES_PIPES__","retries/pipes");
186 define("__INI_SET__","ini_set");
187 define("__PUTENV__","putenv");
188 define("__PUTENV_PATH__","putenv/PATH");
189 // SOME COMMON HEADERS THAT SERVER DETECTS AND REFUSES
190 define("__HTTP_HEADER_1__","User-Agent:");
191 define("__HTTP_HEADER_2__","Accept:");
192 define("__HTTP_HEADER_3__","Host:");
193 define("__HTTP_HEADER_4__","Connection:");
194 // VARIABLES USED IN REPLACEMENTS
195 define("__VARIABLE_NODE__","variables");
196 define("__VARIABLE_STDOUT__","STDOUT");
197 define("__VARIABLE_STDERR__","STDERR");
198 // USED BY CHILDREN MODULE
199 define("__HASH_HOST__","host");
200 define("__HASH_PORT__","port");
201 define("__HASH_NAME__","name");
202 define("__HASH_PID__","pid");
203 define("__HASH_NODE__","node");
204 define("__HASH_ALIAS__","alias");
205 define("__HASH_PIPE__","pipe");
206 define("__CHILD_PID__","pid");
207 define("__CHILD_STDOUT__","stdout");
208 define("__CHILD_STDERR__","stderr");
209 define("__CHILD_TIME__","time");
210 define("__PIPE_CMD__","cmd");
211 define("__PIPE_ARG__","arg");
212 // PROCESS STATUS
213 define("__STATUS_RUN__","run");
214 define("__STATUS_STOP__","stop");
215 define("__STATUS_INIT__","init");
216 // PROCESS COMM
217 define("__WATCHDOG_START__","watchdog_start");
218 define("__WATCHDOG_STOP__","watchdog_stop");
219 define("__PROCESS_START__","process_start");
220 define("__PROCESS_STOP__","process_stop");
221 // BROADCAST
222 define("__BROADCAST_BIND__","0.0.0.0");

```

```

223 define("__BROADCAST_SEND__","255.255.255.255");
224 define("__BROADCAST_HELLO__","HELLO");
225 define("__BROADCAST_ACK__","ACK");
226 // PROCESS LIBRARY
227 define("__PROCESS_PATH__","path");
228 define("__PROCESS_TRACE__","trace");
229 define("__PROCESS_PROCESS__","process");
230 define("__PROCESS_TIMEOUT__","timeout");
231 define("__PROCESS_ONTIMEOUT__","ontimeout");
232 define("__PROCESS_ACTIONS__","actions");
233 define("__PROCESS_ACTION__","action");
234 define("__PROCESS_SHELL__","shell");
235 define("__PROCESS_CHOOSE__","choose");
236 define("__PROCESS_SEND__","send");
237 define("__PROCESS_LOGNODE__","log");
238 define("__PROCESS_WHEN__","when");
239 define("__PROCESS_OTHERWISE__","otherwise");
240 define("__PROCESS_EVAL__","eval");
241 define("__PROCESS_NAME__","name");
242 define("__PROCESS_STDOUT__","stdout");
243 define("__PROCESS_STDERR__","stderr");
244 define("__PROCESS_CONTEXT__","context");
245 define("__PROCESS_FROMITER__","fromiter");
246 define("__PROCESS_EVERYITER__","everyiter");
247 define("__PROCESS_UNTILITER__","untiliter");
248 define("__PROCESS_FROMSEC__","fromsec");
249 define("__PROCESS_EVERYSEC__","everysec");
250 define("__PROCESS_UNTILSEC__","untilsec");
251 define("__PROCESS_CASE__","case");
252 define("__PROCESS_ITERATIONS__","iterations");
253 define("__PROCESS_TIMESTAMP1__","timestamp1");
254 define("__PROCESS_TIMESTAMP2__","timestamp2");
255 define("__PROCESS_PHP__","php");
256 // SOME SCHEDULER KEYS USED TO DEFINE ACTIONS AND CASES
257 define("__SCHEDULER_NODE__","scheduler");
258 define("__SCHEDULER_ACTIONS__","actions");
259 define("__SCHEDULER_HASH__","hash");
260 define("__SCHEDULER_LASTHASH__","lasthash");
261 define("__SCHEDULER_TRIGGERHASH__","triggerhash");
262 // SOME MONITOR KEYS USED TO DEFINE TASKS
263 define("__MONITOR_NODE__","monitor");
264 define("__MONITOR_ACTIONS__","actions");
265 define("__MONITOR_TASK__","task");
266 define("__MONITOR_INTERVAL__","interval");
267 define("__MONITOR_FREQUENCY__","frequency");
268 define("__MONITOR_TIMESTAMP__","timestamp");
269 define("__MONITOR_INIT__","init");
270 // SOME SERVER KEYS USED TO DEFINE SIGNALS
271 define("__SERVER_CHILDHASH__","childhash");
272 define("__SERVER_CMD__","cmd");
273 define("__SERVER_STATUS__","status");
274 ?>

```

Código 48: Contenido del fichero define.php de OpenROCS v2.0

12.7.7. functions.php

Este fichero proporciona las funcionalidades más genéricas y que no se pueden agrupar en otros ficheros.

function file_get_contents_protected(\$file)

Esta función proporciona la funcionalidad de la función de *PHP* llamada *file_get_contents* pero aportando un control adicional contra errores.

function unlink_protected(\$file)

Esta función proporciona la funcionalidad de la función de *PHP* llamada *unlink* pero aportando un control adicional contra errores.

```
function eval_with_vars($eval,$vars="")
```

Esta función proporciona la funcionalidad de la función de *PHP* llamada *eval* pero aportando un control adicional contra errores. Además, permite pasar un array en el parámetro *vars* que son usadas como variables definidas en la evaluación del código pasado por el parámetro *eval*.

```
function str_replace_with_vars($cad,$vars="")
```

Esta función realiza el reemplazo de variables. Es decir, reemplaza las variables que encuentre en la cadena *cad* por sus valores. Estas variables y valores deben encontrarse en el array asociativo que se pasa por el parámetro *vars*.

```
function make_array_vars($array)
```

Esta función generá un array asociativo con variables a partir de un conjunto inicial de variables pasadas por el argumento *array*. Esta función es útil porque permite generar a partir de cadenas de texto, un conjunto de variables para cada conjunto de valores de la variable inicial usando el espacio como separador.

Por ejemplo, si esta función recibe un array como el siguiente:

```
1  Array
2  (
3      [STDOUT] => -rw-rw-r-- 1 sanz sanz 237342 nov 8 14:24 pfc_josep_sanz.tex
4  )
```

Código 49: Ejemplo de un posible valor de STDOUT tras ejecutar un nodo <shell>

Generará una salida como la siguiente:

```
1  Array
2  (
3      [STDOUT] => -rw-rw-r-- 1 sanz sanz 237342 nov 8 14:24 pfc_josep_sanz.tex
4      [STDOUT1] => -rw-rw-r--
5      [STDOUT2] => 1
6      [STDOUT3] => sanz
7      [STDOUT4] => sanz
8      [STDOUT5] => 237342
9      [STDOUT6] => nov
10     [STDOUT7] => 8
11     [STDOUT8] => 14:24
12     [STDOUT9] => pfc_josep_sanz.tex
13  )
```

Código 50: Conjunto de valores para STDOUT* tras ejecutar la función make_array_vars

```
function __config_array_vars_helper($array,$prefix="")
```

Esta función es un helper de la función *config_array_vars*. Tiene la misión de construir, de forma recursiva, los nombres de variables a partir de la información que se obtiene del fichero *variables.xml*.

function config_array_vars()

Esta función genera una lista con variables a partir del contenido del fichero de configuración *variables.xml*. A esta lista se añadirán 3 variables de entorno:

- HOME: Contiene el valor de la variable de entorno HOME que es básicamente, el directorio de trabajo del usuario.
- HOSTNAME: Contiene el nombre de la maquina donde se ejecuta OpenROCS. Es el mismo valor que ejecutar el comando *hostname* en un terminal.
- PATH: Contiene el valor de la variable de entorno PATH que es básicamente, la lista de directorios donde se buscarán binarios para ser ejecutados.

function server_array_vars()

Esta función genera una lista con las variables que tiene el servidor de OpenROCS. Esta lista es usada en los procesos de *monitor* y *scheduler*. Esta lista contiene las variables usando la forma STACK_VARIABLE, de forma que si el STACK=HK tiene una variable llamada TIMESTAMP, en la lista existirá una entrada cuya clave será HK_TIMESTAMP.

Como es una función altamente usada por OpenROCS, se han añadido mejoras como el control del timestamp, es decir, que esta función guarda el resultado y en las siguientes llamadas, sólo preguntará al servidor por los cambios que han habido desde el último instante de tiempo en el cual esta función actualizó sus datos. Esto descarga de transferencia y procesado de datos tanto al servidor como al cliente, pues sólo deben gestionarse los cambios en lugar de generarse la lista nueva. Esto es una optimización bastante importante, puesto la frecuencia en la que esta función se ejecuta es elevada y permite de descargar de computación los dos extremos usados (el cliente, que es quien ejecuta esta función, y el servidor de OpenROCS).

function xml_real_file(\$file)

Esta función, retorna el nombre del fichero pasado por argumento, pero con la prestación de que si existe un fichero en un directorio cuyo nombre es el mismo que el HOSTNAME de la maquina donde se ejecuta, entonces, retorna el segundo. Esto es útil porque OpenROCS puede usar los ficheros de configuración tal como se ha descrito anteriormente, pero en instalaciones donde impliquen varias máquinas, es interesante poder tener todas las máquinas replicadas con el mismo código de configuración y mediante el uso del HOSTNAME, poder especificar ciertas cosas a cada máquina.

function socket_close_protected(&\$socket)

Esta función cierra un socket de forma segura. Esto implica que se controlan los errores que puedan suceder y que además, se llamarán a las siguientes funciones de PHP:

- socket_shutdown
- socket_close

function limpiar_stack(\$stack)

Esta función retorna el nombre del stack sin el sufijo delimitado por el carácter @.

```
function file_exists_protected($file)
```

Esta función proporciona la funcionalidad de la función de *PHP* llamada *file_exists* pero aportando un control adicional contra errores.

```
function addlog_trace($label,$used,$total,$file)
```

Esta función es similar a la función *addlog*, pero con la diferencia que se usa para guardar trazas de la ejecución de OpenROCS. Básicamente, generará una linea de log por cada llamada que contendrá *label* *total=total* *used=used* *percent=percent*, y se guardará en el fichero *file*. El campo *percent* se calcula dentro de esta función a partir de los parámetros *used* y *total*.

12.7.8. gc.php

Este fichero contiene todo el código empleado en la gestión de la basura, o más comúnmente llamado garbage collector.

```
class garbage_collector_helper
```

Esta clase contiene 2 propiedades que son listas para contener los procesos y los ficheros que se están usando.

```
class garbage_collector_helper::function __construct()
```

Este constructor inicializa las 2 propiedades de esta clase.

```
class garbage_collector_helper::function __destruct()
```

Este destructor, tiene por finalidad matar todos los procesos que haya en la lista de procesos y borrar todos los ficheros que haya en la lista de ficheros.

```
class garbage_collector_helper::function add_proc($pid)
```

Este método de la clase tiene por finalidad añadir un proceso, si no existe, a la lista de procesos a controlar.

```
class garbage_collector_helper::function add_file($file)
```

Este método de la clase tiene por finalidad añadir un fichero, si no existe, a la lista de ficheros a controlar.

```
class garbage_collector_helper::function remove_proc($pid)
```

Este método de la clase tiene por finalidad eliminar un proceso, si existe, de la lista de procesos a controlar.

```
class garbage_collector_helper::function remove_file($file)
```

Este método de la clase tiene por finalidad eliminar un fichero, si existe, de la lista de ficheros a controlar.

```
function garbage_collector_init()
```

Esta función crea una instancia a la clase *garbage_collector_helper* para ser usada por el proceso.

```
function garbage_collector_add_proc($pid)
```

Ídem que el método *add_proc* de la clase *garbage_collector_helper*.

```
function garbage_collector_add_file($file)
```

Ídem que el método *add_file* de la clase *garbage_collector_helper*.

```
function garbage_collector_remove_proc($pid)
```

Ídem que el método *remove_proc* de la clase *garbage_collector_helper*.

```
function garbage_collector_remove_file($file)
```

Ídem que el método *remove_file* de la clase *garbage_collector_helper*.

```
function garbage_collector_empty()
```

Ídem que el método *__destruct* de la clase *garbage_collector_helper*.

12.7.9. monitor.php

Este fichero contiene el código que implementa el servicio de monitor. Para ello, usa el siguiente workflow:

- Se programan las señales (SIGTERM, SIGINT, SIGCHLD y SIGUSR1).
- Lee el hash que le envía el proceso padre. Este hash servirá durante toda la vida del proceso para comunicarse con el resto de procesos.
- Inicializa el garbage collector.
- Notifica al padre que se ha activado el servicio.
- Pone el estado del proceso a *stop*.
- define y programa otra función *shutdown_handler*.
- Inicializa la estructura de datos que usará ese monitor.

- Se queda esperando a que alguien ponga el proceso en estado *start*.
- Entra en un bucle infinito para realizar las siguientes tareas:
 - Para cada tarea de ese monitor:
 - Comprobar si debe o no ejecutarse dependiendo de la configuración de la misma

function shutdown_handler()

Esta función, es específica de este servicio y tiene las siguientes tareas asociadas:

- Cerrar el garbage collector.

12.7.10. pipes.php

Este fichero contiene el código que aporta la comunicación entre procesos mediante pipes con nombre.

class pipe

Esta clase implementa la comunicación usando pipes con nombre. Esto es especialmente útil, porque mediante este mecanismo, los procesos pueden compartir información. Esta técnica de comunicación se usa en conjunción con las señales para avisar a los procesos que tienen mensajes pendientes.

Esta clase usa semáforos en la escritura y en la lectura para prevenir problemas de concurrencia.

class pipe::function __construct()

El constructor de esta clase, tiene por objetivo inicializar las 4 variables que se usan para transmitir datos entre procesos:

- Un fichero para el intercambio de información
- Un hash que se usa como firma en la comunicación
- 2 semáforos (1 para escritura y otro para lectura).

class pipe::function write(\$data,\$sync=true)

Este método de la clase tiene por objetivo realizar la escritura del contenido del parámetro *data* en la pipe. Se dispone de un segundo parámetro opcional *sync* que permite que las escrituras sean o no síncronas. La diferencia entre una escritura síncrona y asíncrona es que en la primera, se espera a que se lea el mensaje mientras que en la segunda, únicamente se escribe el mensaje sin esperar que sea leído por el proceso que debe leerlo.

class pipe::function read()

Este método de la clase tiene el objetivo de leer el mensaje que el método *write* realiza sobre la pipe. La lectura es síncrona y finaliza cuando se ha recibido un mensaje completo.

12.7.11. process.php

Este fichero contiene el código usado en OpenROCS para evaluar y procesar los nodos definidos en los procesos *monitor* y *scheduler*. Se puede decir que este fichero contiene el código más importante de OpenROCS, pues el objetivo de este software es procesar las tareas de los monitores y ejecutar el workflow definido en los schedulers. Tanto los procesos *monitor* como los procesos *scheduler* usan este código en su totalidad.

```
function process(&$select)
```

Esta función es la función principal de este módulo y su objetivo es la de procesar el nodo principal de un *monitor* o *scheduler* mediante el uso de dos funciones adicionales, `__process_select` y `__process_choose`. El workflow de la ejecución sigue el siguiente diagrama:

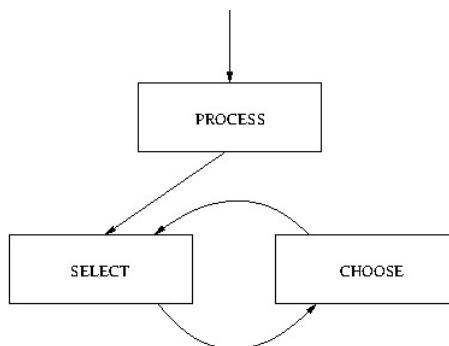


Figura 98: Workflow de la función process

Uno de los objetivos de este módulo es la de ejecutar comandos controlando los siguientes aspectos del proceso:

- Monitorizar si el proceso se está ejecutando o si ya ha finalizado.
- Monitorizar si el proceso ha excedido el tiempo máximo de ejecución (en caso de definir el nodo `<timeout>`).

Pero para ello, el proceso debe ejecutarse de forma asíncrona y así, permitir que el proceso principal no quede bloqueado por el proceso hijo y siga ejecutándose para permitir procesar señales de otros procesos, comprobar los timeouts y en definitiva, sin perder el control del proceso principal.

Todo esto implica que la función `__process_select`, cuando ejecute un nodo de tipo `<shell>`, deberá lanzar el comando y retornar el contexto actual de OpenROCS para que este pueda ser restaurado cuando la ejecución del proceso finalice. Como se puede ver, este concepto es idéntico al usado por los actuales sistemas operativos, que cuando deciden poner otro proceso en ejecución, deben guardar el contexto del proceso para que posteriormente, lo puedan restaurar y continuar la ejecución en el mismo punto donde se encontraba justo antes de parar la ejecución del proceso.

Para conseguir este efecto, la función *process* implementa partes de esta funcionalidad:

- Cuando es ejecutada esta función, se comprueba si existe un contexto guardado en la estructura de datos.
- En caso de que no exista, se llama a la función `__process_select` sin contexto de restauración.

- En caso de que sí exista:
 - Si el proceso no existe, se llama a la función `_process_select` con el contexto guardado para que se restaure la máquina de estados en el punto en que se empezó la ejecución del proceso. Esto permitirá que OpenROCS continúe en el siguiente nodo después del comando `<shell>`.
 - Si el proceso ha excedido el tiempo máximo de ejecución, se para el proceso (mediante las señales SIGTERM y SIGKILL) y se continua la ejecución de la máquina de estados mediante la restauración del contexto guardado.
 - En cualquier otro caso, no se hace nada porque el proceso se está ejecutando y no se puede hacer nada más.

function `_process_select(&$select,$stdout,$stderr,&$context)`

Esta función, tiene como objetivo el procesar los nodos de tipo `<action>`, `<shell>`, `<php>`, `<choose>`, `<send>` y `<log>`. Al igual que la función principal de este módulo, implementa un workflow que permite restaurar el contexto para volver a tener todas las variables con el mismo valor y mantener así, la secuencialidad de la ejecución de la configuración de los ficheros XML de los monitores y schedulers.

function `_process_choose(&$choose,$stdout,$stderr,&$context)`

Esta función, tiene como objetivo el procesar los nodos de tipo `<when>` y `<otherwise>`. Al igual que la función principal de este módulo, implementa un workflow que permite restaurar el contexto para volver a tener todas las variables con el mismo valor y mantener así, la secuencialidad de la ejecución de la configuración de los ficheros XML de los monitores y schedulers.

function `_process_getaction($name)`

Esta función es un helper usado en las funciones:

- `_process_select`
- `_process_select_reset`
- `_process_select_check`

Permite dado un parámetro `name`, retornar el array que representa el XML de un fichero. En caso de usar el parámetro `name` con un valor del tipo `fichero[action]`, entonces además, buscará el nodo de tipo `<action>` cuyo `<name>` sea el valor de `action`.

function `_process_timeout($select,$from)`

Esta función es un helper que permite dado un nodo, buscar si existe el nodo `<timeout>` y `<ontimeout>`.

function `process_reset(&$select)`

Esta función se usa para inicializar las variables internas usadas por OpenROCS como son los contadores y timestamps. El argumento `select` es el nodo principal de un `monitor` o `scheduler`.

```
function __process_select_reset(&$select)
```

Esta función es un helper de la función *process_select* y únicamente hace llamadas recursivas a las funciones *__process_select_reset* y *__process_choose_reset*.

```
function __process_choose_reset(&$choose)
```

Esta función es un helper de la función *process_select* y únicamente hace llamadas recursivas a la función *__process_select_reset*

```
function __process_select_check($select,$file)
```

Esta función es un helper de las funciones *check_monitor* y *check_scheduler*. Únicamente hace llamadas recursivas a las funciones *__process_select_check* y *__process_choose_check* para validar que los valores de los nodos son correctos y coherentes.

```
function __process_choose_check(&$choose,$file)
```

Esta función es un helper de la función *__process_select_check* y únicamente hace llamadas recursivas a la función *__process_select_check* para validar que los valores de los nodos son correctos y coherentes.

12.7.12. scheduler.php

Este fichero contiene el código que implementa el servicio de scheduler. Para ello, usa el siguiente workflow:

- Se programan las señales (SIGTERM, SIGINT, SIGCHLD y SIGUSR1).
- Lee el hash que le envía el proceso padre. Este hash servirá durante toda la vida del proceso para comunicarse con el resto de procesos.
- Inicializa el garbage collector.
- Notifica al padre que se ha activado el servicio.
- Pone el estado del proceso a *stop*.
- define y programa otra función *shutdown_handler*.
- Inicializa la estructura de datos que usará ese scheduler.
- Se queda esperando a que alguien ponga el proceso en estado *start*.
- Entra en un bucle infinito para realizar las siguientes tareas:
 - Comprueba si alguna de las variables definidas como *hash* a cambiado de valor.
 - En caso de que si que cambie alguna variable:
 - Ejecuta la configuración asociada a ese scheduler.

```
function shutdown_handler()
```

Esta función, es específica de este servicio y tiene las siguientes tareas asociadas:

- Cerrar el garbage collector.

12.7.13. server.php

Este fichero contiene el código que implementa el servicio de broadcast. Para ello, usa el siguiente workflow:

- Se programan las señales (SIGTERM, SIGINT, SIGCHLD y SIGUSR1).
- Lee el hash que le envía el proceso padre. Este hash servirá durante toda la vida del proceso para comunicarse con el resto de procesos.
- Inicializa el garbage collector.
- define y programa otra función shutdown_handler.
- Crea un servidor TCP/IP.
- En caso de error, lo notifica al proceso padre y finaliza la ejecución.
- Si todo va bien, notifica al padre que se ha activado el servicio.
- Entra en un bucle infinito hasta que se destruya el socket para realizar las siguientes tareas:
 - Atender las peticiones del servidor TCP/IP:
 - start SERVICE: continua el SERVICE.
 - stop SERVICE: pausa el SERVICE.
 - status: chequea el estado del servidor.
 - status SERVICE: chequea el estado del SERVICE.
 - get: retorna una lista con todos los stacks presentes.
 - get STACK: retorna una lista con todas las variables del STACK.
 - get STACK KEY: retorna el valor de la variable KEY del STACK.
 - get TIMESTAMP: retorna una lista con los stacks que se han modificado desde el TIMESTAMP.
 - get STACK TIMESTAMP: retorna una lista con los datos modificados del STACK desde el TIMESTAMP.
 - add/create STACK: crea el STACK si no existe.
 - add/create STACK KEY: añade la variable KEY al STACK.
 - add/create STACK KEY=VALUE: añade la variable KEY con su valor VALUE al STACK.
 - update/set STACK KEY=VALUE: actualiza en el STACK la variable KEY con el nuevo valor VALUE.
 - remove/delete STACK: borra el STACK sólo si existe y esta vacío.
 - remove/delete STACK KEY: borra la variable KEY del STACK si existe.
 - log MESSAGE: añade el MESSAGE al fichero de user.log.
 - exit/quit/bye: cierra la conexión.
 - crc32: genera el código CRC32 del último comando ejecutado del servidor para el cliente que lo solicita.
 - Comprobar si hay que cerrar conexiones abiertas por timeout.
 - Si hay conexiones que han superado el tiempo máximo de inactividad, se cierran.

- Comprobar si hay señales pendientes de procesar por parte del servidor hacia otros servicios.
 - Si hay señales pendientes, se envía la siguiente (de una en una hasta que finalice).
- Si es necesario, se genera un report con la estadística de uso del servidor (para tareas de depuración).

function shutdown_handler()

Esta función, es específica de este servicio y tiene las siguientes tareas asociadas:

- Cerrar todos los sockets activos del proceso.
- Cerrar el garbage collector.

function __server_compat_helper(\$input,\$__data,\$name)

Esta función es un helper de la función *__server_compat*.

function __server_compat(\$input,\$__data,\$name)

Esta función tiene la misión de separar el nombre del stack, nombre de la variable y el valor, a partir del string *input*. Esta función también resuelve el stack en caso de existir con el sufijo de nombre de máquina.

12.7.14. signals.php

Este fichero contiene el código relativo al procesado de señales. También contiene la función que sirve de gestor de watchdog.

function tick_handler(\$update=null)

Esta función se ejecuta de forma periódica cada 100 ticks internos de PHP. Esto es útil para garantizar que de forma periódica se ejecutará la función que gestiona los watchdogs de los procesos de OpenROCS.

El watchdog funciona de una manera muy simple: debe comprobar que todos los procesos involucrados en la instancia de OpenROCS se están ejecutando sin problemas. En caso de que no se detecte un proceso, el watchdog parará la ejecución del script, cosa que desencadenará que OpenROCS pare de forma inmediata. Este comportamiento es el deseado para garantizar en todo momento la integridad del sistema.

function signal_handler(\$signo)

Esta función tiene la misión de procesar las señales que el proceso recibe. La lista de señales que es capaz de procesar es:

- SIGTERM y SIGINT: detienen la ejecución del script inmediatamente.
- SIGCHLD: pone la variable global *signal.handler_pcntl_wait* a 0. Esto sirve de semáforo para saber cuando se ha recibido una señal de un proceso hijo procedente de un fork anterior y pendiente de finalizar.

- SIGUSR1: esta señal tiene 4 posibles connotaciones:
 - WATCHDOG_START: registra la función *tick_handler* mediante la función *register_tick_function*.
 - WATCHDOG_STOP: des-registra la función *tick_handler* mediante la función *unregister_tick_function*.
 - PROCESS_START: pone el flag de *stopped* a 0.
 - PROCESS_STOP: pone el flag de *stopped* a 1.

function is_stopped(\$update=null)

Esta función gestiona el flag de estado del proceso. Básicamente, indica si el proceso está marcado como *start* o *stop*.

12.7.15. stack.php

Este fichero contiene el código que permite la gestión de los stacks. Básicamente, un stack es un contenedor de variables con su valor.

class stack

Esta clase es una clase contenedora de stacks con sus variables y valores. También permite contener el timestamp y la última operación que se ha realizado sobre el stack y sobre cada variable.

class stack::function __construct()

Este método de la clase construye un objeto del tipo stack e inicializa todas las estructuras internas.

class stack::function add(\$key,\$val)

Este método de la clase añade la variable *key* con su valor *val* al stack. Si la variable *key* ya existe en el stack, no se hace nada y se retorna *false*. Si todo es correcto se retorna *true*.

class stack::function update(\$key,\$val)

Este método de la clase actualiza la variable *key* con su valor *val* en el stack. Si la variable *key* no existe en el stack, no se hace nada y se retorna *false*. Si la variable *key* existe y su valor es el mismo que *val*, tampoco se hace nada y se retorna *false*. Si todo es correcto se retorna *true*.

class stack::function remove(\$key)

Este método de la clase borra la variable *key* del stack. Si la variable *key* no existe en el stack, no se hace nada y se retorna *false*. Si todo es correcto se retorna *true*.

class stack::function import(\$array)

Este método de la clase añade los pares *key* y *val* de cada elemento del *array* al stack. Esta función borra cualquier contenido del stack previo.

class stack::function export()

Este método de la clase retorna un array asociativo con todas las variables del stack.

class stack::function exists(\$key)

Este método de la clase retorna *true* si *key* existe en el stack. En caso contrario retornará *false*.

class stack::function count()

Este método de la clase retorna el número de variables que contiene el stack.

class stack::function modified(\$timestamp)

Este método de la clase retorna *true* si alguna de sus variables ha sido modificada posteriormente a la marca de *timestamp*.

class stack::function __modified_by_action(\$timestamp,\$action)

Este método de la clase es un helper para los métodos *added*, *updated* y *removed*.

class stack::function added(\$timestamp)

Este método de la clase retorna la lista de todas las variables que se han añadido posteriormente a la marca de *timestamp*.

class stack::function updated(\$timestamp)

Este método de la clase retorna la lista de todas las variables que se han modificado posteriormente a la marca de *timestamp*.

class stack::function removed(\$timestamp)

Este método de la clase retorna la lista de todas las variables que se han eliminado posteriormente a la marca de *timestamp*.

12.8. Acciones de OpenROCS

12.8.1. crontab.php

Este fichero tiene la funcionalidad de ejecutar el action *start.php* en caso de que el servidor retorne un mensaje indicando de que OpenROCS esta parado. Este action es usado como action de inicio de OpenROCS desde el crontab del sistema.

12.8.2. help.php

Este fichero tiene el mensaje que debe mostrarse en caso de ejecutar el comando *orocs help*.

12.8.3. reload.php

Este fichero contiene el código necesario para hacer un *reload* de OpenROCS. Esta acción implica que deberá guardar todos los stacks, con sus variables y valores, parar OpenROCS, iniciar OpenROCS y volver a crear los stacks con sus variables y valores.

12.8.4. restart.php

Este fichero contiene el código necesario para hacer un *restart* de OpenROCS. Esta acción implica parar OpenROCS y iniciar OpenROCS.

12.8.5. shell.php

Este fichero contiene el código necesario para implementar una shell con funcionalidades como auto-completado en tiempo real de stacks y variables. Esta prestación es especialmente útil para trabajar en modo ingeniería, que es cuando el operador de telescopio esta realizando tareas no habituales y requiere poder parar ciertas partes del sistema para poder testear otras.

12.8.6. start.php

Este fichero contiene el código necesario para hacer un *start* de OpenROCS. Para ello, usará los siguientes 3 ficheros que implementan partes del proceso de *start*.

12.8.7. start0.php

Este fichero contiene el código necesario para implementar la validación de los ficheros de configuración *monitor.xml* y *scheduler.xml*.

12.8.8. start1.php

Este fichero contiene el código necesario para implementar la creación de todos los procesos involucrados en la ejecución de OpenROCS:

- Crear proceso servidor.

- Crear proceso broadcast (si esta configurado que debe usarse broadcast).
- Crear los procesos monitor.
- Crear los procesos scheduler.

12.8.9. start2.php

Este fichero contiene el código necesario para implementar la inicialización de los watchdogs de todos los procesos.

12.8.10. stop.php

Este fichero contiene el código necesario para hacer un *stop* de OpenROCS. Para ello, usará los siguientes 2 ficheros que implementan partes del proceso de *stop*.

12.8.11. stop1.php

Este fichero contiene el código necesario para implementar la parada del *broadcast*, *monitor* y *scheduler*. Una vez conseguidos estos objetivos, parará todos los watchdogs de todos los procesos.

12.8.12. stop2.php

Este fichero contiene el código necesario para enviar las señales SIGTERM a todos los procesos y esperar a que finalicen. En caso de no parar en un determinado tiempo, se repetirá el proceso con la señal SIGKILL.

12.9. Licencia usada

12.9.1. Cabecera OpenROCS v2.0

```

1  /*
2
3   _ _ \ - -- / _ _ \ / _ _ / _ _ \ / _ _ \ / _ _ \ / _ _ \ / _ _ \
4  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
5  | |_ | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
6  \___/ | . . / \___/ | | | | | | | | | | | | | | | | | | | | | | | |
7  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
8
9  OpenROCS: Open Robotic Observatory Control System
10 Copyright (C) 2011 by Institut d'Estudis Espacials de Catalunya (IEEC)
11 More information in http://www.ieec.cat or ieec@ieec.cat
12
13 This program is free software: you can redistribute it and/or modify
14 it under the terms of the GNU General Public License as published by
15 the Free Software Foundation, either version 3 of the License, or
16 (at your option) any later version.
17
18 This program is distributed in the hope that it will be useful,
19 but WITHOUT ANY WARRANTY; without even the implied warranty of
20 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
21 GNU General Public License for more details.
22
23 You should have received a copy of the GNU General Public License
24 along with this program. If not, see <http://www.gnu.org/licenses/>.
25 */

```

Código 51: Cabecera usada por el proyecto OpenROCS v2.0

12.9.2. Cabecera SaltOS 3.1

```
1  /*
2   -----
3   / _ _ \ / _ ' | | _ | / _ \ \_ / _ _ |
4   \_ _ \ / _ ' | | _ | | _ | \_ _ \ \
5   _ _ ) | ( | | | | _ | | _ | _ ) |
6   | _ _ / \_ , _ | | \_ _ \ \_ _ / | _ _ /
7
8 SaltOS: Framework to develop Rich Internet Applications
9 Copyright (C) 2011 by Josep Sanz Campderros
10 More information in http://www.saltos.net or info@saltos.net
11
12 This program is free software: you can redistribute it and/or modify
13 it under the terms of the GNU General Public License as published by
14 the Free Software Foundation, either version 3 of the License, or
15 (at your option) any later version.
16
17 This program is distributed in the hope that it will be useful,
18 but WITHOUT ANY WARRANTY; without even the implied warranty of
19 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
20 GNU General Public License for more details.
21
22 You should have received a copy of the GNU General Public License
23 along with this program. If not, see <http://www.gnu.org/licenses/>.
24 */
```

Código 52: Cabecera usada por el proyecto SaltOS

12.10. Ficheros XML usados en el TJO

12.10.1. config.xml

```
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <!--
3   ---
4   / - \ - -- | --- \ / - \ / --- / --- \
5   | | | | | , \ / - \ , - \| | | | | | | | | |
6   | | | | | | | | | | | | | | | | | | | | | | | |
7   \---/ .---\ \---/ | | | | | | | | | | | | | | | | | |
8   |_|
9
10 OpenROCS: Open Robotic Observatory Control System
11 Copyright (C) 2011-2014 by Institut d'Estudis Espacials de Catalunya (IEEC)
12 More information in http://www.ieec.cat or ieec@ieec.cat
13
14 This program is free software: you can redistribute it and/or modify
15 it under the terms of the GNU General Public License as published by
16 the Free Software Foundation, either version 3 of the License, or
17 (at your option) any later version.
18
19 This program is distributed in the hope that it will be useful,
20 but WITHOUT ANY WARRANTY; without even the implied warranty of
21 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
22 GNU General Public License for more details.
23
24 You should have received a copy of the GNU General Public License
25 along with this program. If not, see <http://www.gnu.org/licenses/>.
26 -->
27 <root>
28   <server>
29     <host>127.0.0.1</host>
30     <port>2323</port>
31     <name>estall</name>
32     <stacks>
33       <stack>HK</stack>
34       <stack>SDP</stack>
35       <stack>SYS</stack>
36     </stacks>
37   </server>
38   <broadcast>
39     <enabled>true</enabled>
40     <port>2424</port>
```

```
1      <discovery>60</discovery>
2      <synchronize>10</synchronize>
3  </broadcast>
4  <debug>
5      <comm>false</comm>
6      <signal>false</signal>
7      <process>false</process>
8      <trace>true</trace>
9      <maxlines>1000</maxlines>
10     <percent>50</percent>
11  </debug>
12  <shell>
13      <history>$HOME/.orocs_history</history>
14      <maxlines>1000</maxlines>
15  </shell>
16  <timeout>
17      <comm>1000000</comm>
18      <childs>1000000</childs>
19      <pipes>3000000</pipes>
20      <wait>5000000</wait>
21      <server>5000000</server>
22      <semaphore>5000000</semaphore>
23  </timeout>
24  <polling>
25      <comm>1000</comm>
26      <childs>1000</childs>
27      <pipes>1000</pipes>
28      <wait>100000</wait>
29      <server>100000</server>
30      <monitor>100000</monitor>
31      <scheduler>100000</scheduler>
32      <broadcast>100000</broadcast>
33  </polling>
34  <retries>
35      <comm>5</comm>
36      <childs>5</childs>
37      <pipes>5</pipes>
38  </retries>
39  <ini_set>
40      <session.bug_compat_42>On</session.bug_compat_42>
41      <register_globals>Off</register_globals>
42      <memory_limit>128M</memory_limit>
43      <max_execution_time>0</max_execution_time>
44      <date.timezone>UTC</date.timezone>
45      <default_charset>UTF-8</default_charset>
46  </ini_set>
47  <putenv>
48      <!--<PATH>/bin:/usr/bin:/usr/local/bin</PATH>-->
49      <LANG>en_US.UTF-8</LANG>
50  </putenv>
51 </root>
```

12.10.2. variables.xml

```
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <!--
3   /_ \ _ _ \ _ _ _ \ _ _ _ \ _ _ _ \ _ _ _ \
4   | | | | | | | | | | | | | | | | | | | | |
5   | | | | | | | | | | | | | | | | | | | | |
6   | | | | | | | | | | | | | | | | | | | | |
7   \_/_| .__\_\_/_| | | | | | | | | | | | |
8   | | | | | | | | | | | | | | | | | | | |
9
10 OpenROCS: Open Robotic Observatory Control System
11 Copyright (C) 2011-2014 by Institut d'Estudis Espacials de Catalunya (IEEC)
12 More information in http://www.ieec.cat or iec@ieec.cat
13
14 This program is free software: you can redistribute it and/or modify
15 it under the terms of the GNU General Public License as published by
16 the Free Software Foundation, either version 3 of the License, or
17 (at your option) any later version.
18
19 This program is distributed in the hope that it will be useful,
20 but WITHOUT ANY WARRANTY; without even the implied warranty of
21 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
22 GNU General Public License for more details.
23
24 You should have received a copy of the GNU General Public License
```

```

25 along with this program. If not, see <http://www.gnu.org/licenses/>.
26 -->
27 <root>
28   <variables>
29     <!-- IPs -->
30     <IP>
31       <SERVER>SERVER_IP</SERVER>
32       <NAS>NAS_IP</NAS>
33       <TELESCOPE>TELESCOPE_IP</TELESCOPE>
34       <REDUNDANT>REDUNDANT_IP</REDUNDANT>
35       <METEO>METEO_IP</METEO>
36       <ROUTER>ROUTER_IP</ROUTER>
37       <DESKTOP>DESKTOP_IP</DESKTOP>
38       <PDU>
39         <NO>PDU_NO_IP</NO>
40         <N1>PDU_N1_IP</N1>
41         <N2>PDU_N2_IP</N2>
42       </PDU>
43       <OTHERS>www.oadm.cat</OTHERS>
44     </IP>
45     <!-- File System paths and files -->
46   <FS>
47     <PID>
48       <TELRUN>/usr/local/telescope/comm/telrun.pid</TELRUN>
49       <CSIMCD>/usr/local/telescope/comm/csimcd.pid</CSIMCD>
50       <TELESCOPED>/usr/local/telescope/comm/telescoped.pid</TELESCOPED>
51       <CAMERAD>/usr/local/telescope/comm/camerad.pid</CAMERAD>
52       <WXD_READ>/usr/local/telescope/comm/wxd.pid</WXD_READ>
53       <GPSD>/usr/local/telescope/comm/gpsd.pid</GPSD>
54       <RUND>
55         <TELRUN>/usr/local/telescope/comm/rund.telrun.pid</TELRUN>
56         <CSIMCD>/usr/local/telescope/comm/rund.csimcd.pid</CSIMCD>
57         <TELESCOPED>/usr/local/telescope/comm/rund.telescoped.pid</TELESCOPED>
58         <CAMERAD>/usr/local/telescope/comm/rund.camerad.pid</CAMERAD>
59         <WXD_WRITE>/usr/local/telescope/comm/rund.wxd.php.pid</WXD_WRITE>
60         <WXD_READ>/usr/local/telescope/comm/rund.wxd.pid</WXD_READ>
61         <GPSD>/usr/local/telescope/comm/rund.gpsd.pid</GPSD>
62       </RUND>
63     </PID>
64     <FIFO>
65       <CAMERA_IN>/usr/local/telescope/comm/Camera.in</CAMERA_IN>
66       <CAMERA_OUT>/usr/local/telescope/comm/Camera.out</CAMERA_OUT>
67       <TEL_IN>/usr/local/telescope/comm/Tel.in</TEL_IN>
68       <TEL_OUT>/usr/local/telescope/comm/Tel.out</TEL_OUT>
69       <COVER_IN>/usr/local/telescope/comm/Cover.in</COVER_IN>
70       <COVER_OUT>/usr/local/telescope/comm/Cover.out</COVER_OUT>
71       <FOCUS_IN>/usr/local/telescope/comm/Focus.in</FOCUS_IN>
72       <FOCUS_OUT>/usr/local/telescope/comm/Focus.out</FOCUS_OUT>
73       <FILTER_IN>/usr/local/telescope/comm/Filter.in</FILTER_IN>
74       <FILTER_OUT>/usr/local/telescope/comm/Filter.out</FILTER_OUT>
75       <DOME_IN>/usr/local/telescope/comm/Dome.in</DOME_IN>
76       <DOME_OUT>/usr/local/telescope/comm/Dome.out</DOME_OUT>
77       <LIGHTS_IN>/usr/local/telescope/comm/Lights.in</LIGHTS_IN>
78       <LIGHTS_OUT>/usr/local/telescope/comm/Lights.out</LIGHTS_OUT>
79       <POWERFAIL_IN>/usr/local/telescope/comm/Powerfail.in</POWERFAIL_IN>
80       <POWERFAIL_OUT>/usr/local/telescope/comm/Powerfail.out</POWERFAIL_OUT>
81     </FIFO>
82     <FILE>
83       <TELRUN>/usr/local/telescope/archive/telrun/telrun.sls</TELRUN>
84       <WXD>/usr/local/telescope/archive/logs/wxd.dat</WXD>
85     </FILE>
86     <LOG>
87       <TELESCOPED>/usr/local/telescope/archive/logs/telescoped.log</TELESCOPED>
88       <CAMERAD>/usr/local/telescope/archive/logs/camerad.log</CAMERAD>
89     </LOG>
90   </FS>
91   <!-- Iteration variables -->
92   <ITER>
93     <UPDATE>1</UPDATE>
94     <CHECK>3</CHECK>
95   </ITER>
96   <!-- Time variables -->
97   <TIME>
98     <NOMINAL>10</NOMINAL>
99     <CHECK>5</CHECK>
100    <SHORT>1</SHORT>
101    <RUND>
102      <TELESCOPED>60</TELESCOPED>
103      <CAMERAD>10</CAMERAD>
104    </RUND>
105    <METEO>
106      <CHECK>60</CHECK>
107    </METEO>

```

```

108     </TIME>
109     <!-- Logging level -->
110     <LOG>
111         <NO>Information:</NO>
112         <N1>Warning:</N1>
113         <N2>Minor error:</N2>
114         <N3>Major error:</N3>
115         <N4>Critical error:</N4>
116     </LOG>
117     <!-- PDU status -->
118     <PDU>
119         <ON>1</ON>
120         <OFF>2</OFF>
121     </PDU>
122     <!-- Observatory OIDs -->
123     <OID>
124         <PDU>
125             <N1>.1.3.6.1.4.1.318.1.1.12.3.3.1.1.4.1</N1>
126             <N2>.1.3.6.1.4.1.318.1.1.12.3.3.1.1.4.2</N2>
127             <N3>.1.3.6.1.4.1.318.1.1.12.3.3.1.1.4.3</N3>
128             <N4>.1.3.6.1.4.1.318.1.1.12.3.3.1.1.4.4</N4>
129             <N5>.1.3.6.1.4.1.318.1.1.12.3.3.1.1.4.5</N5>
130             <N6>.1.3.6.1.4.1.318.1.1.12.3.3.1.1.4.6</N6>
131             <N7>.1.3.6.1.4.1.318.1.1.12.3.3.1.1.4.7</N7>
132             <N8>.1.3.6.1.4.1.318.1.1.12.3.3.1.1.4.8</N8>
133         </PDU>
134         <DAVIS>.1.3.6.1.4.1.2021.50.1.4.1.2.5.100.97.118.105.115</DAVIS>
135     </OID>
136     <!-- Ports -->
137     <PORT>
138         <CSIMCD>7623</CSIMCD>
139     </PORT>
140     <!-- Meteorological ranges -->
141     <METEO>
142         <MIN>
143             <TEMPERATURE>-50</TEMPERATURE>
144             <HUMIDITY_DRY>0</HUMIDITY_DRY>
145             <HUMIDITY_WET>90</HUMIDITY_WET>
146             <WIND>0</WIND>
147             <WIND_MEDIUM>13</WIND_MEDIUM>
148             <DIRECTION>0</DIRECTION>
149             <PRESSURE>600</PRESSURE>
150             <RAIN>0</RAIN>
151         </MIN>
152         <MAX>
153             <TEMPERATURE>50</TEMPERATURE>
154             <HUMIDITY>110</HUMIDITY>
155             <WIND_MEDIUM>16</WIND_MEDIUM>
156             <WIND>70</WIND>
157             <DIRECTION>360</DIRECTION>
158             <PRESSURE>1100</PRESSURE>
159             <RAIN>1000</RAIN>
160         </MAX>
161     </METEO>
162 </variables>
163 </root>
```

12.10.3. monitor.xml

```
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <!--
3
4   /_ \ - -- _ _ _ _ | _ \ / _ \ / _ / _ _ |
5   | | | | | _ \ / _ \ | | | | | | | | | | | |
6   | | | | | | | | | | | | | | | | | | | | | |
7   \_ _/_ | . _ / \_ _/_ | | | | | | | | | | | |
8   | | | | | | | | | | | | | | | | | | | | | |
9
10 OpenROCS: Open Robotic Observatory Control System
11 Copyright (C) 2011-2014 by Institut d'Estudis Espacials de Catalunya (IEEC)
12 More information in http://www.ieec.cat or ieec@ieec.cat
13
14 This program is free software: you can redistribute it and/or modify
15 it under the terms of the GNU General Public License as published by
16 the Free Software Foundation, either version 3 of the License, or
17 (at your option) any later version.
18
19 This program is distributed in the hope that it will be useful,
```

```

20 but WITHOUT ANY WARRANTY; without even the implied warranty of
21 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
22 GNU General Public License for more details.
23
24 You should have received a copy of the GNU General Public License
25 along with this program. If not, see <http://www.gnu.org/licenses/>.
26 -->
27 <root>
28     <!-- Monitor: GOAL. Check the existence of SYS_GOAL and SYS_USER variables -->
29     <monitor>
30         <name>goal</name>
31         <task>
32             <send>add SYS_USER=UNKNOWN</send>
33             <choose>
34                 <when>
35                     <eval><! [CDATA [
36                         $SYS_USER == 'UNKNOWN',
37                         ]]></eval>
38                     <send>add SYS_GOAL=STOP</send>
39                 </when>
40                 <otherwise>
41                     <send>add SYS_GOAL=$SYS_USER</send>
42                 </otherwise>
43             </choose>
44             <interval>$TIME_NOMINAL</interval>
45         </task>
46     </monitor>
47     <!-- Monitor: STATUS. Check the existence of the SYS_STATUS variable -->
48     <monitor>
49         <name>status</name>
50         <task>
51             <send>add SYS_STATUS=UNKNOWN</send>
52             <interval>$TIME_NOMINAL</interval>
53         </task>
54     </monitor>
55     <!-- Monitor: NODE. Update variables for those systems with a network -->
56     <!-- connection (router, SAIs, PDUs and computers) -->
57     <monitor>
58         <name>node</name>
59         <!-- Define all the HK_NODE undefined variables -->
60         <!-- Check TCP/IP to router -->
61         <task>
62             <shell>ping -c 3 -i 1 -W 1 $IP_ROUTER | grep received</shell>
63             <choose>
64                 <when>
65                     <eval>$STDOUT4>0</eval>
66                     <send>update HK_NODE_ROUTER_PING_STATUS=GOOD</send>
67                 </when>
68                 <otherwise>
69                     <send>update HK_NODE_ROUTER_PING_STATUS=ERROR</send>
70                 </otherwise>
71             </choose>
72             <interval>$TIME_NOMINAL</interval>
73         </task>
74         <!-- Check TCP/IP to localhost (TELESCOPE) -->
75         <task>
76             <shell>ping -c 3 -i 1 -W 1 $IP_TELESCOPE | grep received</shell>
77             <interval>$TIME_NOMINAL</interval>
78             <choose>
79                 <when>
80                     <eval>$STDOUT4>0</eval>
81                     <send>update HK_NODE_TELESCOPE_PING_STATUS=GOOD</send>
82                 </when>
83                 <otherwise>
84                     <send>update HK_NODE_TELESCOPE_PING_STATUS=ERROR</send>
85                 </otherwise>
86             </choose>
87         </task>
88         <!-- Check TCP/IP to local server (NAS) -->
89         <task>
90             <shell>ping -c 3 -i 1 -W 1 $IP_NAS | grep received</shell>
91             <interval>$TIME_NOMINAL</interval>
92             <choose>
93                 <when>
94                     <eval>$STDOUT4>0</eval>
95                     <send>update HK_NODE_NAS_PING_STATUS=GOOD</send>
96                 </when>
97                 <otherwise>
98                     <send>update HK_NODE_NAS_PING_STATUS=ERROR</send>
99                 </otherwise>
100            </choose>
101        </task>
102        <!-- Check TCP/IP to redundant system (REDUNDANT) -->

```

```

103      <task>
104          <shell>ping -c 3 -i 1 -W 1 $IP_REDUNDANT | grep received</shell>
105          <interval>$TIME_NOMINAL</interval>
106          <choose>
107              <when>
108                  <eval>$STDOUT4>0</eval>
109                  <send>update HK_NODE_REDUNDANT_PING_STATUS=GOOD</send>
110              </when>
111              <otherwise>
112                  <send>update HK_NODE_REDUNDANT_PING_STATUS=ERROR</send>
113              </otherwise>
114          </choose>
115      </task>
116      <!-- Check TCP/IP to meteorological server (METEO) -->
117      <task>
118          <shell>ping -c 3 -i 1 -W 1 $IP_METEO | grep received</shell>
119          <interval>$TIME_NOMINAL</interval>
120          <choose>
121              <when>
122                  <eval>$STDOUT4>0</eval>
123                  <send>update HK_NODE_METEO_PING_STATUS=GOOD</send>
124              </when>
125              <otherwise>
126                  <send>update HK_NODE_METEO_PING_STATUS=ERROR</send>
127              </otherwise>
128          </choose>
129      </task>
130      <!-- Check TCP/IP to local terminal (DESKTOP) -->
131      <task>
132          <shell>ping -c 3 -i 1 -W 1 $IP_DESKTOP | grep received</shell>
133          <interval>$TIME_NOMINAL</interval>
134          <choose>
135              <when>
136                  <eval>$STDOUT4>0</eval>
137                  <send>update HK_NODE_DESKTOP_PING_STATUS=GOOD</send>
138              </when>
139              <otherwise>
140                  <send>update HK_NODE_DESKTOP_PING_STATUS=ERROR</send>
141              </otherwise>
142          </choose>
143      </task>
144      <!-- Check TCP/IP to control room PDU (PDU_NO) -->
145      <task>
146          <shell>ping -c 3 -i 1 -W 1 $IP_PDU_NO | grep received</shell>
147          <interval>$TIME_NOMINAL</interval>
148          <choose>
149              <when>
150                  <eval>$STDOUT4>0</eval>
151                  <send>update HK_NODE_PDU_NO_PING_STATUS=GOOD</send>
152              </when>
153              <otherwise>
154                  <send>update HK_NODE_PDU_NO_PING_STATUS=ERROR</send>
155              </otherwise>
156          </choose>
157      </task>
158      <!-- Check TCP/IP to telescope/instrument PDU (PDU_N1) -->
159      <task>
160          <shell>ping -c 3 -i 1 -W 1 $IP_PDU_N1 | grep received</shell>
161          <interval>$TIME_NOMINAL</interval>
162          <choose>
163              <when>
164                  <eval>$STDOUT4>0</eval>
165                  <send>update HK_NODE_PDU_N1_PING_STATUS=GOOD</send>
166              </when>
167              <otherwise>
168                  <send>update HK_NODE_PDU_N1_PING_STATUS=ERROR</send>
169              </otherwise>
170          </choose>
171      </task>
172      <!-- Check TCP/IP to communications PDU (PDU_N2) -->
173      <task>
174          <shell>ping -c 3 -i 1 -W 1 $IP_PDU_N2 | grep received</shell>
175          <interval>$TIME_NOMINAL</interval>
176          <choose>
177              <when>
178                  <eval>$STDOUT4>0</eval>
179                  <send>update HK_NODE_PDU_N2_PING_STATUS=GOOD</send>
180              </when>
181              <otherwise>
182                  <send>update HK_NODE_PDU_N2_PING_STATUS=ERROR</send>
183              </otherwise>
184          </choose>
185      </task>

```

```

186      <!-- Check router power supply -->
187      <task>
188          <shell>snmpget -v1 -c public $IP_PDU_N2 $OID_PDU_N3</shell>
189          <choose>
190              <when>
191                  <eval>$STDOUT4===$PDU_ON</eval>
192                  <send>update HK_NODE_ROUTER_POWER_STATUS=ON</send>
193              </when>
194              <when>
195                  <eval>$STDOUT4===$PDU_OFF</eval>
196                  <send>update HK_NODE_ROUTER_POWER_STATUS=OFF</send>
197              </when>
198              <otherwise>
199                  <send>update HK_NODE_ROUTER_POWER_STATUS=ERROR</send>
200              </otherwise>
201          </choose>
202          <interval>$TIME_NOMINAL</interval>
203      </task>
204      <!-- Check TELESCOPE power supply -->
205      <task>
206          <shell>snmpget -v1 -c public $IP_PDU_NO $OID_PDU_N1</shell>
207          <choose>
208              <when>
209                  <eval>$STDOUT4===$PDU_ON</eval>
210                  <send>update HK_NODE_TELESCOPE_POWER_STATUS=ON</send>
211              </when>
212              <when>
213                  <eval>$STDOUT4===$PDU_OFF</eval>
214                  <send>update HK_NODE_TELESCOPE_POWER_STATUS=OFF</send>
215              </when>
216              <otherwise>
217                  <send>update HK_NODE_TELESCOPE_POWER_STATUS=ERROR</send>
218              </otherwise>
219          </choose>
220          <interval>$TIME_NOMINAL</interval>
221      </task>
222      <!-- Check NAS power supply (two power supplies) -->
223      <task>
224          <shell>snmpget -v1 -c public $IP_PDU_N2 $OID_PDU_N1</shell>
225          <choose>
226              <when>
227                  <eval>$STDOUT4===$PDU_ON</eval>
228                  <send>update HK_NODE_NAS_POWER_N1_STATUS=ON</send>
229              </when>
230              <when>
231                  <eval>$STDOUT4===$PDU_OFF</eval>
232                  <send>update HK_NODE_NAS_POWER_N1_STATUS=OFF</send>
233              </when>
234              <otherwise>
235                  <send>update HK_NODE_NAS_POWER_N1_STATUS=ERROR</send>
236              </otherwise>
237          </choose>
238          <interval>$TIME_NOMINAL</interval>
239      </task>
240      <task>
241          <shell>snmpget -v1 -c public $IP_PDU_N2 $OID_PDU_N2</shell>
242          <choose>
243              <when>
244                  <eval>$STDOUT4===$PDU_ON</eval>
245                  <send>update HK_NODE_NAS_POWER_N2_STATUS=ON</send>
246              </when>
247              <when>
248                  <eval>$STDOUT4===$PDU_OFF</eval>
249                  <send>update HK_NODE_NAS_POWER_N2_STATUS=OFF</send>
250              </when>
251              <otherwise>
252                  <send>update HK_NODE_NAS_POWER_N2_STATUS=ERROR</send>
253              </otherwise>
254          </choose>
255          <interval>$TIME_NOMINAL</interval>
256      </task>
257      <!-- Check REDUNDANT power supply -->
258      <task>
259          <shell>snmpget -v1 -c public $IP_PDU_NO $OID_PDU_N3</shell>
260          <choose>
261              <when>
262                  <eval>$STDOUT4===$PDU_ON</eval>
263                  <send>update HK_NODE_REDUNDANT_POWER_STATUS=ON</send>
264              </when>
265              <when>
266                  <eval>$STDOUT4===$PDU_OFF</eval>
267                  <send>update HK_NODE_REDUNDANT_POWER_STATUS=OFF</send>
268              </when>

```

```

269          <otherwise>
270              <send>update HK_NODE_REDUNDANT_POWER_STATUS=ERROR</send>
271          </otherwise>
272      </choose>
273      <interval>$TIME_NOMINAL</interval>
274  </task>
275  <!-- Check METEO power supply -->
276  <task>
277      <shell>snmpget -v1 -c public $IP_PDU_NO $OID_PDU_N4</shell>
278      <choose>
279          <when>
280              <eval>$STDOUT4===$PDU_ON</eval>
281              <send>update HK_NODE_METEO_POWER_STATUS=ON</send>
282          </when>
283          <when>
284              <eval>$STDOUT4===$PDU_OFF</eval>
285              <send>update HK_NODE_METEO_POWER_STATUS=OFF</send>
286          </when>
287          <otherwise>
288              <send>update HK_NODE_METEO_POWER_STATUS=ERROR</send>
289          </otherwise>
290      </choose>
291      <interval>$TIME_NOMINAL</interval>
292  </task>
293  <!-- Check DESKTOP power supply -->
294  <task>
295      <shell>snmpget -v1 -c public $IP_PDU_NO $OID_PDU_N2</shell>
296      <choose>
297          <when>
298              <eval>$STDOUT4===$PDU_ON</eval>
299              <send>update HK_NODE_DESKTOP_POWER_STATUS=ON</send>
300          </when>
301          <when>
302              <eval>$STDOUT4===$PDU_OFF</eval>
303              <send>update HK_NODE_DESKTOP_POWER_STATUS=OFF</send>
304          </when>
305          <otherwise>
306              <send>update HK_NODE_DESKTOP_POWER_STATUS=ERROR</send>
307          </otherwise>
308      </choose>
309      <interval>$TIME_NOMINAL</interval>
310  </task>
311 </monitor>
312 <!-- Monitor: DEVICE. Update variables for those systems without a network -->
313 <!-- connection (fiber-links, sensors, etc.) -->
314 <monitor>
315     <name>device</name>
316     <!-- Check UdL radiofrequency fiber link power supply -->
317     <task>
318         <shell>snmpget -v1 -c public $IP_PDU_N2 $OID_PDU_N4</shell>
319         <choose>
320             <when>
321                 <eval>$STDOUT4===$PDU_ON</eval>
322                 <send>update HK_DEVICE_UDL_FIBER_POWER_STATUS=ON</send>
323             </when>
324             <when>
325                 <eval>$STDOUT4===$PDU_OFF</eval>
326                 <send>update HK_DEVICE_UDL_FIBER_POWER_STATUS=OFF</send>
327             </when>
328             <otherwise>
329                 <send>update HK_DEVICE_UDL_FIBER_POWER_STATUS=ERROR</send>
330             </otherwise>
331         </choose>
332         <interval>$TIME_NOMINAL</interval>
333     </task>
334     <!-- Check communications tower power supply -->
335     <task>
336         <send>add HK_DEVICE_TOWER_POWER_STATUS=ERROR</send>
337         <shell>snmpget -v1 -c public $IP_PDU_N2 $OID_PDU_N5</shell>
338         <choose>
339             <when>
340                 <eval>$STDOUT4===$PDU_ON</eval>
341                 <send>update HK_DEVICE_TOWER_POWER_STATUS=ON</send>
342             </when>
343             <when>
344                 <eval>$STDOUT4===$PDU_OFF</eval>
345                 <send>update HK_DEVICE_TOWER_POWER_STATUS=OFF</send>
346             </when>
347             <otherwise>
348                 <send>update HK_DEVICE_TOWER_POWER_STATUS=ERROR</send>
349             </otherwise>
350         </choose>
351         <interval>$TIME_NOMINAL</interval>

```

```

352      </task>
353  </monitor>
354  <!-- Monitor: METEO. Update variables related with time and -->
355  <!--               meteorological information -->
356  <monitor>
357      <name>meteo</name>
358      <!-- Show the changing timestamp to prove that stack is being refreshed -->
359  <task>
360      <php>time()</php>
361      <interval>$TIME_SHORT</interval>
362      <send>update SDP_METEO_TIMESTAMP=$STDOUT1</send>
363  </task>
364  <!-- Check wxd.php daemon -->
365  <task>
366      <send>add HK_METEO_WXD_WRITE_GOAL=UNKNOWN</send>
367      <send>add HK_METEO_WXD_WRITE_STATUS=ERROR</send>
368      <choose>
369          <when>
370              <eval><! [CDATA [
371                  $HK_METEO_WXD_WRITE_STATUS != 'STARTING' , &&
372                  $HK_METEO_WXD_WRITE_STATUS != 'STOPPING',
373              ]]></eval>
374              <action>meteo.xml[wxd_write_status]</action>
375          </when>
376      </choose>
377      <interval>$TIME_NOMINAL</interval>
378  </task>
379  <!-- Check wxd daemon -->
380  <task>
381      <send>add HK_METEO_WXD_READ_GOAL=UNKNOWN</send>
382      <send>add HK_METEO_WXD_READ_STATUS=ERROR</send>
383      <choose>
384          <when>
385              <eval><! [CDATA [
386                  $HK_METEO_WXD_READ_STATUS != 'STARTING' , &&
387                  $HK_METEO_WXD_READ_STATUS != 'STOPPING',
388              ]]></eval>
389              <action>meteo.xml[wxd_read_status]</action>
390          </when>
391      </choose>
392      <interval>$TIME_NOMINAL</interval>
393  </task>
394  <!-- Check gpsd daemon -->
395  <task>
396      <send>add HK_METEO_GPSD_GOAL=UNKNOWN</send>
397      <send>add HK_METEO_GPSD_STATUS=ERROR</send>
398      <choose>
399          <when>
400              <eval><! [CDATA [
401                  $HK_METEO_GPSD_STATUS != 'STARTING' , &&
402                  $HK_METEO_GPSD_STATUS != 'STOPPING',
403              ]]></eval>
404              <action>meteo.xml[gpsd_status]</action>
405          </when>
406      </choose>
407      <interval>$TIME_NOMINAL</interval>
408  </task>
409  <!-- Check meteorological sensors power supply -->
410  <task>
411      <send>add HK_METEO_SENSORS_POWER_STATUS=ERROR</send>
412      <shell>snmpget -v1 -c public $IP_PDU_NO $OID_PDU_N8</shell>
413      <choose>
414          <when>
415              <eval>$STDOUT4===$PDU_ON</eval>
416              <send>update HK_METEO_SENSORS_POWER_STATUS=ON</send>
417          </when>
418          <when>
419              <eval>$STDOUT4===$PDU_OFF</eval>
420              <send>update HK_METEO_SENSORS_POWER_STATUS=OFF</send>
421          </when>
422          <otherwise>
423              <send>update HK_METEO_SENSORS_POWER_STATUS=ERROR</send>
424          </otherwise>
425      </choose>
426      <interval>$TIME_NOMINAL</interval>
427  </task>
428  <!-- Retrieve meteorological data from Davis weather station -->
429  <task>
430      <shell>snmpwalk -c public -v 1 $IP_METEO $OID_DAVIS | cut -f 2,2 -d ''</shell>
431      <!-- Parse the resulting string to retrieve each data value -->
432      <choose>
433          <when>
434              <eval><! [CDATA [strpos($STDOUT , "DAVIS") !== FALSE]]></eval>

```

```

435      <!-- Control room temperature -->
436      <choose>
437          <when>
438              <eval><! [CDATA [
439                  is_numeric($STDOUT3) &&
440                  $STDOUT3 >= $METEO_MIN_TEMPERATURE &&
441                  $STDOUT3 <= $METEO_MAX_TEMPERATURE &&
442                  $STDOUT4 - $SDP_METEO_DAVIS_TEMPERATURE_INDOOR_TIME > 0.0
443              ]]></eval>
444              <send>update HK_METEO_DAVIS_TEMPERATURE_INDOOR_STATUS=GOOD</send>
445              <send>update SDP_METEO_DAVIS_TEMPERATURE_INDOOR_VALUE=$STDOUT3 </<-
446                  send>
447                  <send>update SDP_METEO_DAVIS_TEMPERATURE_INDOOR_TIME=$STDOUT4 </<-
448                      send>
449              </when>
450              <when>
451                  <eval><! [CDATA [
452                      !isset($SDP_METEO_DAVIS_TEMPERATURE_INDOOR_TIME) || 
453                      $STDOUT4 - $SDP_METEO_DAVIS_TEMPERATURE_INDOOR_TIME <= 0.0
454                  ]]></eval>
455                  <send>update HK_METEO_DAVIS_TEMPERATURE_INDOOR_STATUS=<-
456                      OUTDATED_ERROR </send>
457                  <send>add SDP_METEO_DAVIS_TEMPERATURE_INDOOR_TIME=$STDOUT4 </send>
458                  <fromsec>$TIME_METEO_CHECK </fromsec>
459              </when>
460              <otherwise>
461                  <send>update HK_METEO_DAVIS_TEMPERATURE_INDOOR_STATUS=VALUE_ERROR <-
462                      </send>
463                  <send>update SDP_METEO_DAVIS_TEMPERATURE_INDOOR_VALUE=$STDOUT3 </<-
464                      send>
465                  <send>update SDP_METEO_DAVIS_TEMPERATURE_INDOOR_TIME=$STDOUT4 </<-
466                      send>
467              </otherwise>
468          </choose>
469          <!-- Control room humidity -->
470          <choose>
471              <when>
472                  <eval><! [CDATA [
473                      is_numeric($STDOUT7) &&
474                      $STDOUT7 >= $METEO_MIN_HUMIDITY_DRY &&
475                      $STDOUT7 < $METEO_MIN_HUMIDITY_WET &&
476                      $STDOUT8 - $SDP_METEO_DAVIS_HUMIDITY_INDOOR_TIME > 0.0
477                  ]]></eval>
478                  <send>update HK_METEO_DAVIS_HUMIDITY_INDOOR_STATUS=DRY</send>
479                  <send>update SDP_METEO_DAVIS_HUMIDITY_INDOOR_VALUE=$STDOUT7 </send>-
480                  >
481                  <send>update SDP_METEO_DAVIS_HUMIDITY_INDOOR_TIME=$STDOUT8 </send>
482              </when>
483              <when>
484                  <eval><! [CDATA [
485                      is_numeric($STDOUT7) &&
486                      $STDOUT7 >= $METEO_MIN_HUMIDITY_WET &&
487                      $STDOUT7 < $METEO_MAX_HUMIDITY &&
488                      $STDOUT8 - $SDP_METEO_DAVIS_HUMIDITY_INDOOR_TIME > 0.0
489                  ]]></eval>
490                  <send>update HK_METEO_DAVIS_HUMIDITY_INDOOR_STATUS=WET</send>
491                  <send>update SDP_METEO_DAVIS_HUMIDITY_INDOOR_VALUE=$STDOUT7 </send>-
492                  >
493                  <send>update SDP_METEO_DAVIS_HUMIDITY_INDOOR_TIME=$STDOUT8 </send>
494              </when>
495              <when>
496                  <eval><! [CDATA [
497                      !isset($SDP_METEO_DAVIS_HUMIDITY_INDOOR_TIME) || 
498                      $STDOUT8 - $SDP_METEO_DAVIS_HUMIDITY_INDOOR_TIME <= 0.0
499                  ]]></eval>
500                  <send>update HK_METEO_DAVIS_HUMIDITY_INDOOR_STATUS=OUTDATED_ERROR <-
501                      </send>
502                  <send>add SDP_METEO_DAVIS_HUMIDITY_INDOOR_TIME=$STDOUT8 </send>
503                  <fromsec>$TIME_METEO_CHECK </fromsec>
504              </when>
505              <otherwise>
506                  <send>update HK_METEO_DAVIS_HUMIDITY_INDOOR_STATUS=VALUE_ERROR <-
507                      </send>
508                  <send>update SDP_METEO_DAVIS_HUMIDITY_INDOOR_VALUE=$STDOUT7 </send>-
509                  >
510                  <send>update SDP_METEO_DAVIS_HUMIDITY_INDOOR_TIME=$STDOUT8 </send>
511              </otherwise>
512          </choose>
513          <!-- TJO temperature -->
514          <choose>
515              <when>
516                  <eval><! [CDATA [
517                      is_numeric($STDOUT11) &&

```

```

507             $STDOUT11 >= $METEO_MIN_TEMPERATURE &&
508             $STDOUT11 <= $METEO_MAX_TEMPERATURE &&
509             $STDOUT12 - $SDP_METEO_DAVIS_TEMPERATURE_TJO_TIME > 0.0
510         ]]></eval>
511         <send>update HK_METEO_DAVIS_TEMPERATURE_TJO_STATUS=GOOD</send>
512         <send>update SDP_METEO_DAVIS_TEMPERATURE_TJO_VALUE=$STDOUT11</→
513             send>
514             <send>update SDP_METEO_DAVIS_TEMPERATURE_TJO_TIME=$STDOUT12</send→
515             >
516     </when>
517     <when>
518         <eval><![CDATA [
519             !isset($SDP_METEO_DAVIS_TEMPERATURE_TJO_TIME) ||
520             $STDOUT12 - $SDP_METEO_DAVIS_TEMPERATURE_TJO_TIME <= 0.0
521         ]]></eval>
522         <send>update HK_METEO_DAVIS_TEMPERATURE_TJO_STATUS=OUTDATED_ERROR←
523             </send>
524         <send>add SDP_METEO_DAVIS_TEMPERATURE_TJO_TIME=$STDOUT12</send>
525             <fromsec>$TIME_METEO_CHECK</fromsec>
526     </when>
527     <otherwise>
528         <send>update HK_METEO_DAVIS_TEMPERATURE_TJO_STATUS=VALUE_ERROR</→
529             send>
530         <send>update SDP_METEO_DAVIS_TEMPERATURE_TJO_VALUE=$STDOUT11</→
531             send>
532         <send>update SDP_METEO_DAVIS_TEMPERATURE_TJO_TIME=$STDOUT12</send→
533             >
534     </otherwise>
535     </choose>
536     <!-- TJO humidity -->
537     <choose>
538         <when>
539             <eval><![CDATA [
540                 is_numeric($STDOUT15) &&
541                 $STDOUT15 >= $METEO_MIN_HUMIDITY_DRY &&
542                 $STDOUT15 < $METEO_MIN_HUMIDITY_WET &&
543                 $STDOUT16 - $SDP_METEO_DAVIS_HUMIDITY_TJO_TIME > 0.0
544             ]]></eval>
545             <send>update HK_METEO_DAVIS_HUMIDITY_TJO_STATUS=DRY</send>
546             <send>update SDP_METEO_DAVIS_HUMIDITY_TJO_VALUE=$STDOUT15</send>
547             <send>update SDP_METEO_DAVIS_HUMIDITY_TJO_TIME=$STDOUT16</send>
548         </when>
549         <when>
550             <eval><![CDATA [
551                 is_numeric($STDOUT15) &&
552                 $STDOUT15 >= $METEO_MIN_HUMIDITY_WET &&
553                 $STDOUT15 < $METEO_MAX_HUMIDITY &&
554                 $STDOUT16 - $SDP_METEO_DAVIS_HUMIDITY_TJO_TIME > 0.0
555             ]]></eval>
556             <send>update HK_METEO_DAVIS_HUMIDITY_TJO_STATUS=WET</send>
557             <send>update SDP_METEO_DAVIS_HUMIDITY_TJO_VALUE=$STDOUT15</send>
558             <send>update SDP_METEO_DAVIS_HUMIDITY_TJO_TIME=$STDOUT16</send>
559         </when>
560         <when>
561             <eval><![CDATA [
562                 !isset($SDP_METEO_DAVIS_HUMIDITY_TJO_TIME) ||
563                 $STDOUT16 - $SDP_METEO_DAVIS_HUMIDITY_TJO_TIME <= 0.0
564             ]]></eval>
565             <send>update HK_METEO_DAVIS_HUMIDITY_TJO_STATUS=OUTDATED_ERROR</→
566                 send>
567             <send>add SDP_METEO_DAVIS_HUMIDITY_TJO_TIME=$STDOUT16</send>
568                 <fromsec>$TIME_METEO_CHECK</fromsec>
569     </when>
570     <otherwise>
571         <send>update HK_METEO_DAVIS_HUMIDITY_TJO_STATUS=VALUE_ERROR</send→
572         >
573         <send>update SDP_METEO_DAVIS_HUMIDITY_TJO_VALUE=$STDOUT15</send>
574         <send>update SDP_METEO_DAVIS_HUMIDITY_TJO_TIME=$STDOUT16</send>
575     </otherwise>
576     </choose>
577     <!-- Outdoor temperature -->
578     <choose>
579         <when>

```

```

580             <send>update SDP_METEO_DAVIS_TEMPERATURE_OUTDOOR_TIME=$STDOUT20</→
581                     send>
582         </when>
583     <when>
584         <eval><![CDATA [
585             !isset($SDP_METEO_DAVIS_TEMPERATURE_OUTDOOR_TIME) ||
586             $STDOUT20-$SDP_METEO_DAVIS_TEMPERATURE_OUTDOOR_TIME <=0.0
587         ]]></eval>
588         <send>update HK_METEO_DAVIS_TEMPERATURE_OUTDOOR_STATUS=<→
589                         OUTDATED_ERROR</send>
590         <send>add SDP_METEO_DAVIS_TEMPERATURE_OUTDOOR_TIME=$STDOUT20</→
591                     send>
592         <fromsec>$TIME_METEO_CHECK</fromsec>
593     </when>
594     <otherwise>
595         <send>update HK_METEO_DAVIS_TEMPERATURE_OUTDOOR_STATUS=<→
596                         VALUE_ERROR</send>
597         <send>update SDP_METEO_DAVIS_TEMPERATURE_OUTDOOR_VALUE=$STDOUT19<→
598                         </send>
599         <send>update SDP_METEO_DAVIS_TEMPERATURE_OUTDOOR_TIME=$STDOUT20</→
600                     send>
601     </otherwise>
602   </choose>
603   <!-- Outdoor humidity -->
604   <choose>
605     <when>
606         <eval><![CDATA [
607             is_numeric($STDOUT23) &&
608             $STDOUT23 >=$METEO_MIN_HUMIDITY_DRY &&
609             $STDOUT23 <$METEO_MIN_HUMIDITY_WET &&
610             $STDOUT24-$SDP_METEO_DAVIS_HUMIDITY_OUTDOOR_TIME >0.0
611         ]]></eval>
612         <send>update HK_METEO_DAVIS_HUMIDITY_OUTDOOR_STATUS=DRY</send>
613         <send>update SDP_METEO_DAVIS_HUMIDITY_OUTDOOR_VALUE=$STDOUT23</→
614                         send>
615         <send>update SDP_METEO_DAVIS_HUMIDITY_OUTDOOR_TIME=$STDOUT24</→
616                         send>
617     </when>
618     <when>
619         <eval><![CDATA [
620             is_numeric($STDOUT23) &&
621             $STDOUT23 >=$METEO_MIN_HUMIDITY_WET &&
622             $STDOUT23 <$METEO_MAX_HUMIDITY &&
623             $STDOUT24-$SDP_METEO_DAVIS_HUMIDITY_OUTDOOR_TIME >0.0
624         ]]></eval>
625         <send>update HK_METEO_DAVIS_HUMIDITY_OUTDOOR_STATUS=WET</send>
626         <send>update SDP_METEO_DAVIS_HUMIDITY_OUTDOOR_VALUE=$STDOUT23</→
627                         send>
628         <send>update SDP_METEO_DAVIS_HUMIDITY_OUTDOOR_TIME=$STDOUT24</→
629                         send>
630     </when>
631     <when>
632         <eval><![CDATA [
633             !isset($SDP_METEO_DAVIS_HUMIDITY_OUTDOOR_TIME) ||
634             $STDOUT24-$SDP_METEO_DAVIS_HUMIDITY_OUTDOOR_TIME <=0.0
635         ]]></eval>
636         <send>update HK_METEO_DAVIS_HUMIDITY_OUTDOOR_STATUS=<→
637                         OUTDATED_ERROR</send>
638         <send>add SDP_METEO_DAVIS_HUMIDITY_OUTDOOR_TIME=$STDOUT24</send>
639     </when>
640     <otherwise>
641         <send>update HK_METEO_DAVIS_HUMIDITY_OUTDOOR_STATUS=VALUE_ERROR</→
642                     send>
643         <send>update SDP_METEO_DAVIS_HUMIDITY_OUTDOOR_VALUE=$STDOUT23</→
644                         send>
645         <send>update SDP_METEO_DAVIS_HUMIDITY_OUTDOOR_TIME=$STDOUT24</→
646                         send>
647     </otherwise>
648   </choose>
649   <!-- Wind speed -->
650   <choose>
651     <when>
652         <eval><![CDATA [
653             is_numeric($STDOUT27) &&
654             $STDOUT27 >=$METEO_MIN_WIND &&
655             $STDOUT27 <$METEO_MIN_WIND_MEDIUM &&
656             $STDOUT28-$SDP_METEO_DAVIS_WIND_SPEED_TIME >0.0
657         ]]></eval>
658         <send>update HK_METEO_DAVIS_WIND_SPEED_STATUS=LIGHT</send>
659         <send>update SDP_METEO_DAVIS_WIND_SPEED_VALUE=$STDOUT27</send>
660         <send>update SDP_METEO_DAVIS_WIND_SPEED_TIME=$STDOUT28</send>
661     </when>

```

```

649      <when>
650          <eval><!CDATA[
651              is_numeric($STDOUT27) &&
652              $STDOUT27 >= $METEO_MIN_WIND_MEDIUM &&
653              $STDOUT27 <=$METEO_MAX_WIND_MEDIUM &&
654              $STDOUT28-$SDP_METEO_DAVIS_WIND_SPEED_TIME >0.0
655          ]]></eval>
656          <send>update HK_METEO_DAVIS_WIND_SPEED_STATUS=MEDIUM</send>
657          <send>update SDP_METEO_DAVIS_WIND_SPEED_VALUE=$STDOUT27</send>
658          <send>update SDP_METEO_DAVIS_WIND_SPEED_TIME=$STDOUT28</send>
659      </when>
660      <when>
661          <eval><!CDATA[
662              is_numeric($STDOUT27) &&
663              $STDOUT27 >= $METEO_MAX_WIND_MEDIUM &&
664              $STDOUT27 <=$METEO_MAX_WIND &&
665              $STDOUT28-$SDP_METEO_DAVIS_WIND_SPEED_TIME >0.0
666          ]]></eval>
667          <send>update HK_METEO_DAVIS_WIND_SPEED_STATUS=STRONG</send>
668          <send>update SDP_METEO_DAVIS_WIND_SPEED_VALUE=$STDOUT27</send>
669          <send>update SDP_METEO_DAVIS_WIND_SPEED_TIME=$STDOUT28</send>
670      </when>
671      <when>
672          <eval><!CDATA[
673              !isset($SDP_METEO_DAVIS_WIND_SPEED_TIME) ||
674              $STDOUT28-$SDP_METEO_DAVIS_WIND_SPEED_TIME <=0.0
675          ]]></eval>
676          <send>update HK_METEO_DAVIS_WIND_SPEED_STATUS=OUTDATED_ERROR</-->
677          <send>add SDP_METEO_DAVIS_WIND_SPEED_TIME=$STDOUT28</send>
678          <fromsec>$TIME_METEO_CHECK</fromsec>
679      </when>
680      <otherwise>
681          <send>update HK_METEO_DAVIS_WIND_SPEED_STATUS=VALUE_ERROR</send>
682          <send>update SDP_METEO_DAVIS_WIND_SPEED_VALUE=$STDOUT27</send>
683          <send>update SDP_METEO_DAVIS_WIND_SPEED_TIME=$STDOUT28</send>
684      </otherwise>
685  </choose>
686  <!-- Wind direction -->
687  <choose>
688      <when>
689          <eval><!CDATA[
690              is_numeric($STDOUT31) &&
691              $STDOUT31 >= $METEO_MIN_DIRECTION &&
692              $STDOUT31 <= $METEO_MAX_DIRECTION &&
693              $STDOUT32-$SDP_METEO_DAVIS_WIND_DIRECTION_TIME >0.0
694          ]]></eval>
695          <send>update HK_METEO_DAVIS_WIND_DIRECTION_STATUS=GOOD</send>
696          <send>update SDP_METEO_DAVIS_WIND_DIRECTION_VALUE=$STDOUT31</send>-
697          <send>update SDP_METEO_DAVIS_WIND_DIRECTION_TIME=$STDOUT32</send>
698      </when>
699      <when>
700          <eval><!CDATA[
701              !isset($SDP_METEO_DAVIS_WIND_DIRECTION_TIME) ||
702              $STDOUT32-$SDP_METEO_DAVIS_WIND_DIRECTION_TIME <=0.0
703          ]]></eval>
704          <send>update HK_METEO_DAVIS_WIND_DIRECTION_STATUS=OUTDATED_ERROR</-->
705          <send>add SDP_METEO_DAVIS_WIND_DIRECTION_TIME=$STDOUT32</send>
706          <fromsec>$TIME_METEO_CHECK</fromsec>
707      </when>
708      <otherwise>
709          <send>update HK_METEO_DAVIS_WIND_DIRECTION_STATUS=VALUE_ERROR</-->
710          <send>update SDP_METEO_DAVIS_WIND_DIRECTION_VALUE=$STDOUT31</send>-
711          <send>update SDP_METEO_DAVIS_WIND_DIRECTION_TIME=$STDOUT32</send>
712      </otherwise>
713  </choose>
714  <!-- Pressure -->
715  <choose>
716      <when>
717          <eval><!CDATA[
718              is_numeric($STDOUT35) &&
719              $STDOUT35 >= $METEO_MIN_PRESSURE &&
720              $STDOUT35 <= $METEO_MAX_PRESSURE &&
721              $STDOUT36-$SDP_METEO_DAVIS_PRESSURE_TIME >0.0
722          ]]></eval>
723          <send>update HK_METEO_DAVIS_PRESSURE_STATUS=GOOD</send>
724          <send>update SDP_METEO_DAVIS_PRESSURE_VALUE=$STDOUT35</send>
725          <send>update SDP_METEO_DAVIS_PRESSURE_TIME=$STDOUT36</send>
726      </when>

```

```

727
728         <when>
729             <eval><![CDATA[
730                 !isset($SDP_METEO_DAVIS_PRESSURE_TIME) ||
731                 $STDOUT36 - $SDP_METEO_DAVIS_PRESSURE_TIME <= 0.0
732             ]]></eval>
733             <send>update HK_METEO_DAVIS_PRESSURE_STATUS=OUTDATED_ERROR</send>
734             <send>update SDP_METEO_DAVIS_PRESSURE_TIME=$STDOUT36</send>
735             <fromsec>$TIME_METEO_CHECK</fromsec>
736         </when>
737         <otherwise>
738             <send>update HK_METEO_DAVIS_PRESSURE_STATUS=VALUE_ERROR</send>
739             <send>update SDP_METEO_DAVIS_PRESSURE_VALUE=$STDOUT35</send>
740             <send>update SDP_METEO_DAVIS_PRESSURE_TIME=$STDOUT36</send>
741         </otherwise>
742     </choose>
743     <!-- Rain rate -->
744     <choose>
745         <when>
746             <eval><![CDATA[
747                 is_numeric($STDOUT39) &&
748                 $STDOUT39 == $METEO_MIN_RAIN &&
749                 $STDOUT40 - $SDP_METEO_DAVIS_RAIN_TIME > 0.0
750             ]]></eval>
751             <send>update HK_METEO_DAVIS_RAIN_STATUS=DRY</send>
752             <send>update SDP_METEO_DAVIS_RAIN_VALUE=$STDOUT39</send>
753             <send>update SDP_METEO_DAVIS_RAIN_TIME=$STDOUT40</send>
754         </when>
755         <when>
756             <eval><![CDATA[
757                 is_numeric($STDOUT39) &&
758                 $STDOUT39 <= $METEO_MAX_RAIN &&
759                 $STDOUT40 - $SDP_METEO_DAVIS_RAIN_TIME > 0.0
760             ]]></eval>
761             <send>update HK_METEO_DAVIS_RAIN_STATUS=RAINING</send>
762             <send>update SDP_METEO_DAVIS_RAIN_VALUE=$STDOUT39</send>
763             <send>update SDP_METEO_DAVIS_RAIN_TIME=$STDOUT40</send>
764         </when>
765         <when>
766             <eval><![CDATA[
767                 !isset($SDP_METEO_DAVIS_RAIN_TIME) ||
768                 $STDOUT40 - $SDP_METEO_DAVIS_RAIN_TIME <= 0.0
769             ]]></eval>
770             <send>update HK_METEO_DAVIS_RAIN_STATUS=OUTDATED_ERROR</send>
771             <send>add SDP_METEO_DAVIS_RAIN_TIME=$STDOUT40</send>
772             <fromsec>$TIME_METEO_CHECK</fromsec>
773         </when>
774         <otherwise>
775             <send>update HK_METEO_DAVIS_RAIN_STATUS=VALUE_ERROR</send>
776             <send>update SDP_METEO_DAVIS_RAIN_VALUE=$STDOUT39</send>
777             <send>update SDP_METEO_DAVIS_RAIN_TIME=$STDOUT40</send>
778         </otherwise>
779     </choose>
780   </when>
781   <otherwise>
782       <send>update HK_METEO_DAVIS_TEMPERATURE_INDOOR_STATUS=GET_ERROR</send>
783       <send>update HK_METEO_DAVIS_HUMIDITY_INDOOR_STATUS=GET_ERROR</send>
784       <send>update HK_METEO_DAVIS_TEMPERATURE_TJO_STATUS=GET_ERROR</send>
785       <send>update HK_METEO_DAVIS_HUMIDITY_TJO_STATUS=GET_ERROR</send>
786       <send>update HK_METEO_DAVIS_TEMPERATURE_OUTDOOR_STATUS=GET_ERROR</send>
787       <send>update HK_METEO_DAVIS_HUMIDITY_OUTDOOR_STATUS=GET_ERROR</send>
788       <send>update HK_METEO_DAVIS_WIND_SPEED_STATUS=GET_ERROR</send>
789       <send>update HK_METEO_DAVIS_WIND_DIRECTION_STATUS=GET_ERROR</send>
790       <send>update HK_METEO_DAVIS_PRESSURE_STATUS=GET_ERROR</send>
791       <send>update HK_METEO_DAVIS_RAIN_STATUS=GET_ERROR</send>
792       <fromsec>$TIME_METEO_CHECK</fromsec>
793   </otherwise>
794 </choose>
795   <interval>$TIME_NOMINAL</interval>
796 </task>
797 </monitor>
798 <!-- Monitor: TJO. Update variables for TJO -->
799 <monitor>
800     <name>tjo</name>
801     <!-- Check TJO power supply: 5V -->
802     <task>
803         <send>add HK_TJO_5V_POWER_GOAL=UNKNOWN</send>
804         <send>add HK_TJO_5V_POWER_STATUS=ERROR</send>
805         <choose>
806             <when>
807                 <eval><![CDATA[
808                     $HK_TJO_5V_POWER_STATUS != 'STARTING' &&
809                     $HK_TJO_5V_POWER_STATUS != 'STOPPING',

```

```

810                                ]]></eval>
811                                <action>tjo.xml[5v_pdu_status]</action>
812                            </when>
813                        </choose>
814                        <interval>$TIME_NOMINAL</interval>
815                    </task>
816                    <!-- Check TJO power supply: 12V -->
817                    <task>
818                        <send>add HK_TJO_12V_POWER_GOAL=UNKNOWN</send>
819                        <send>add HK_TJO_12V_POWER_STATUS=ERROR</send>
820                        <choose>
821                            <when>
822                                <eval><! [CDATA [
823                                    $HK_TJO_12V_POWER_STATUS != 'STARTING' &&
824                                    $HK_TJO_12V_POWER_STATUS != 'STOPPING',
825                                ]]></eval>
826                                <action>tjo.xml[12v_pdu_status]</action>
827                            </when>
828                        </choose>
829                        <interval>$TIME_NOMINAL</interval>
830                    </task>
831                    <!-- Check TJO power supply: 24V -->
832                    <task>
833                        <send>add HK_TJO_24V_POWER_GOAL=UNKNOWN</send>
834                        <send>add HK_TJO_24V_POWER_STATUS=ERROR</send>
835                        <choose>
836                            <when>
837                                <eval><! [CDATA [
838                                    $HK_TJO_24V_POWER_STATUS != 'STARTING' &&
839                                    $HK_TJO_24V_POWER_STATUS != 'STOPPING',
840                                ]]></eval>
841                                <action>tjo.xml[24v_pdu_status]</action>
842                            </when>
843                        </choose>
844                        <interval>$TIME_NOMINAL</interval>
845                    </task>
846                    <!-- Check TJO power supply: 80V -->
847                    <task>
848                        <send>add HK_TJO_80V_POWER_GOAL=UNKNOWN</send>
849                        <send>add HK_TJO_80V_POWER_STATUS=ERROR</send>
850                        <choose>
851                            <when>
852                                <eval><! [CDATA [
853                                    $HK_TJO_80V_POWER_STATUS != 'STARTING' &&
854                                    $HK_TJO_80V_POWER_STATUS != 'STOPPING',
855                                ]]></eval>
856                                <action>tjo.xml[80v_pdu_status]</action>
857                            </when>
858                        </choose>
859                        <interval>$TIME_NOMINAL</interval>
860                    </task>
861                    <!-- Check telescoped daemon -->
862                    <task>
863                        <send>add HK_TJO_TELESCOPED_GOAL=UNKNOWN</send>
864                        <send>add HK_TJO_TELESCOPED_STATUS=ERROR</send>
865                        <choose>
866                            <when>
867                                <eval><! [CDATA [
868                                    $HK_TJO_TELESCOPED_STATUS != 'STARTING' &&
869                                    $HK_TJO_TELESCOPED_STATUS != 'STOPPING',
870                                ]]></eval>
871                                <action>tjo.xml[telescoped_status]</action>
872                            </when>
873                        </choose>
874                        <interval>$TIME_NOMINAL</interval>
875                    </task>
876                    <!-- Check csimcd daemon -->
877                    <task>
878                        <send>add HK_TJO_CSIMCD_GOAL=UNKNOWN</send>
879                        <send>add HK_TJO_CSIMCD_STATUS=ERROR</send>
880                        <choose>
881                            <when>
882                                <eval><! [CDATA [
883                                    $HK_TJO_CSIMCD_STATUS != 'STARTING' &&
884                                    $HK_TJO_CSIMCD_STATUS != 'STOPPING',
885                                ]]></eval>
886                                <action>tjo.xml[csimcd_status]</action>
887                            </when>
888                        </choose>
889                        <interval>$TIME_NOMINAL</interval>
890                    </task>
891                </monitor>
892                <!-- Monitor: MEIA. Update variables for MEIA -->

```

```

893     <monitor>
894         <name>meia</name>
895         <!-- Check camerad daemon -->
896         <task>
897             <send>add HK_MEIA_CAMERAD_GOAL=UNKNOWN</send>
898             <send>add HK_MEIA_CAMERAD_STATUS=ERROR</send>
899             <choose>
900                 <when>
901                     <eval><! [CDATA [
902                         $HK_MEIA_CAMERAD_STATUS != 'STARTING' &&
903                         $HK_MEIA_CAMERAD_STATUS != 'STOPPING',
904                     ]]></eval>
905                     <action>meia.xml[camerad_status]</action>
906                 </when>
907             </choose>
908             <interval>$TIME_NOMINAL</interval>
909         </task>
910         <!-- Check MEIA CCD temperature -->
911         <task>
912             <send>add SDP_MEIA_TEMP_VALUE=0</send>
913             <send>add HK_MEIA_TEMP_GOAL=UNKNOWN</send>
914             <send>add HK_MEIA_TEMP_STATUS=ERROR</send>
915             <send>add HK_MEIA_CAMERAD_STATUS=ERROR</send>
916             <shell>ps x | grep -c "telrun$$"</shell>
917             <choose>
918                 <when>
919                     <eval><! [CDATA [
920                         $STDOUT==0 &&
921                         $HK_MEIA_CAMERAD_STATUS != 'STARTING' &&
922                         $HK_MEIA_CAMERAD_STATUS != 'STOPPING',
923                     ]]></eval>
924                     <action>meia.xml[temp_status]</action>
925                 </when>
926             </choose>
927             <interval>$TIME_NOMINAL</interval>
928         </task>
929     </monitor>
930     <!-- Monitor: DOME. Update variables for dome -->
931     <monitor>
932         <name>dome</name>
933         <!-- Check dome circuits power -->
934         <task>
935             <send>add HK_DOME_CIRCUITS_POWER_GOAL=UNKNOWN</send>
936             <send>add HK_DOME_CIRCUITS_POWER_STATUS=ERROR</send>
937             <choose>
938                 <when>
939                     <eval><! [CDATA [
940                         $HK_DOME_CIRCUITS_POWER_STATUS != 'STARTING' &&
941                         $HK_DOME_CIRCUITS_POWER_STATUS != 'STOPPING',
942                     ]]></eval>
943                     <action>dome.xml[circuits_pdu_status]</action>
944                 </when>
945             </choose>
946             <interval>$TIME_NOMINAL</interval>
947         </task>
948         <!-- Check dome light power -->
949         <task>
950             <send>add HK_DOME_LIGHT_POWER_GOAL=UNKNOWN</send>
951             <send>add HK_DOME_LIGHT_POWER_STATUS=ERROR</send>
952             <choose>
953                 <when>
954                     <eval><! [CDATA [
955                         $HK_DOME_LIGHT_POWER_STATUS != 'STARTING' &&
956                         $HK_DOME_LIGHT_POWER_STATUS != 'STOPPING',
957                     ]]></eval>
958                     <action>dome.xml[light_pdu_status]</action>
959                 </when>
960             </choose>
961             <interval>$TIME_NOMINAL</interval>
962         </task>
963         <!-- Check dome flats power -->
964         <task>
965             <send>add HK_DOME_FLATS_POWER_STATUS=ERROR</send>
966             <choose>
967                 <when>
968                     <eval><! [CDATA [
969                         $HK_DOME_FLATS_POWER_STATUS != 'STARTING' &&
970                         $HK_DOME_FLATS_POWER_STATUS != 'STOPPING',
971                     ]]></eval>
972                     <action>dome.xml[flats_pdu_status]</action>
973                 </when>
974             </choose>
975             <interval>$TIME_NOMINAL</interval>

```

```

976      </task>
977  </monitor>
978  <!-- Monitor: TELRUN. Update variables for telrun -->
979  <monitor>
980      <name>telrun</name>
981      <!-- Check the length of telrun.sls file -->
982      <task>
983          <shell>wc -l $FS_FILE_TELRUN</shell>
984          <timeout>$TIME_SHORT</timeout>
985          <choose>
986              <when>
987                  <eval>$STDERR==''</eval>
988                  <send>update SDP_TELRUN_NLINES_VALUE=$STDOUT1</send>
989                  <choose>
990                      <when>
991                          <eval>$STDOUT1>0</eval>
992                          <send>update HK_TELRUN_NLINES_STATUS=FULL</send>
993                      </when>
994                      <otherwise>
995                          <send>update HK_TELRUN_NLINES_STATUS=EMPTY</send>
996                      </otherwise>
997                  </choose>
998              </when>
999              <otherwise>
1000                  <send>update SDP_TELRUN_NLINES_VALUE=0</send>
1001                  <send>update HK_TELRUN_NLINES_STATUS=ERROR</send>
1002              </otherwise>
1003          </choose>
1004          <interval>$TIME_NOMINAL</interval>
1005      </task>
1006      <!-- Check the number of observations pending -->
1007      <task>
1008          <shell>grep "status: N" -A 7 $FS_FILE_TELRUN | grep " Target ID:" | grep -vc " ↵
Calibration," </shell>
1009          <timeout>$TIME_SHORT</timeout>
1010          <choose>
1011              <when>
1012                  <eval>$STDERR==''</eval>
1013                  <send>update SDP_TELRUN_NOBS_VALUE=$STDOUT1</send>
1014                  <choose>
1015                      <when>
1016                          <eval>$STDOUT1>0</eval>
1017                          <send>update HK_TELRUN_NOBS_STATUS=PENDING</send>
1018                      </when>
1019                      <otherwise>
1020                          <send>update HK_TELRUN_NOBS_STATUS=DONE</send>
1021                      </otherwise>
1022                  </choose>
1023              </when>
1024              <otherwise>
1025                  <send>update SDP_TELRUN_NOBS_VALUE=0</send>
1026                  <send>update HK_TELRUN_NOBS_STATUS=ERROR</send>
1027              </otherwise>
1028          </choose>
1029          <interval>$TIME_NOMINAL</interval>
1030      </task>
1031      <!-- Check the number of calibrations pending -->
1032      <task>
1033          <shell>grep "status: N" -A 7 $FS_FILE_TELRUN | grep " Target ID:" | grep -c " ↵
Calibration," </shell>
1034          <timeout>$TIME_SHORT</timeout>
1035          <choose>
1036              <when>
1037                  <eval>$STDERR==''</eval>
1038                  <send>update SDP_TELRUN_NCAL_VALUE=$STDOUT1</send>
1039                  <choose>
1040                      <when>
1041                          <eval>$STDOUT1>0</eval>
1042                          <send>update HK_TELRUN_NCAL_STATUS=PENDING</send>
1043                      </when>
1044                      <otherwise>
1045                          <send>update HK_TELRUN_NCAL_STATUS=DONE</send>
1046                      </otherwise>
1047                  </choose>
1048              </when>
1049              <otherwise>
1050                  <send>update SDP_TELRUN_NCAL_VALUE=0</send>
1051                  <send>update HK_TELRUN_NCAL_STATUS=ERROR</send>
1052              </otherwise>
1053          </choose>
1054          <interval>$TIME_NOMINAL</interval>
1055      </task>
1056      <!-- Check telrun daemon -->

```

```

1057      <task>
1058          <send>add HK_TELRUN_DAEMON_GOAL=UNKNOWN</send>
1059          <send>add HK_TELRUN_DAEMON_STATUS=ERROR</send>
1060          <choose>
1061              <when>
1062                  <eval><![CDATA[
1063                      $HK_TELRUN_DAEMON_STATUS != 'STARTING' &&
1064                      $HK_TELRUN_DAEMON_STATUS != 'STOPPING',
1065                  ]]></eval>
1066                  <action>telrun.xml[daemon_status]</action>
1067              </when>
1068          </choose>
1069          <interval>$TIME_NOMINAL</interval>
1070      </task>
1071  </monitor>
1072 </root>

```

12.10.4. scheduler.xml

```

1  <?xml version="1.0" encoding="UTF-8" ?>
2  <!--
3
4  / _ \ - -- \ / --- - -- | _ \ / _ \ / --- / --- | -- _ \ / _ \ / _ \
5  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
6  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
7  \---/ | .--/ \---| | | | | | | | | | | | | | | | | | | | | | | | | |
8  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
9
10 OpenROCS: Open Robotic Observatory Control System
11 Copyright (C) 2011-2014 by Institut d'Estudis Espacials de Catalunya (IEEC)
12 More information in http://www.ieec.cat or ieec@ieec.cat
13
14 This program is free software: you can redistribute it and/or modify
15 it under the terms of the GNU General Public License as published by
16 the Free Software Foundation, either version 3 of the License, or
17 (at your option) any later version.
18
19 This program is distributed in the hope that it will be useful,
20 but WITHOUT ANY WARRANTY; without even the implied warranty of
21 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
22 GNU General Public License for more details.
23
24 You should have received a copy of the GNU General Public License
25 along with this program. If not, see <http://www.gnu.org/licenses/>.
26 -->
27 <root>
28     <!-- Scheduler: INIT. Define OpenROCS start-up actions -->
29     <scheduler>
30         <name>init</name>
31         <hash><variable></variable></hash>
32         <log>$LOG_NO OpenROCS started</log>
33         <!-- Recover the system variables from broadcast (not implemented yet) -->
34     </scheduler>
35     <!-- Scheduler: GOAL. Define de SYS_GOAL -->
36     <scheduler>
37         <name>goal</name>
38         <hash>
39             <variable>HK_TELRUN_NCAL_STATUS</variable>
40             <variable>HK_TELRUN_NOBS_STATUS</variable>
41             <variable>HK_TELRUN_NLINES_STATUS</variable>
42             <variable>SYS_USER</variable>
43             <variable>SYS_GOAL</variable>
44         </hash>
45         <log>$LOG_NO Entering GOAL scheduler</log>
46         <!-- First, define the GOAL based on USER will -->
47         <choose>
48             <when>
49                 <eval><![CDATA[
50                     $SYS_USER=='STOP' ||
51                     $SYS_USER=='START' ||
52                     $SYS_USER=='INIT' ||
53                     $SYS_USER=='THERMAL',
54                 ]]></eval>
55                 <send>update SYS_GOAL=$SYS_USER</send>
56                 <log>$LOG_NO Setting SYS_GOAL=$SYS_USER</log>
57             </when>
58             <when>
59                 <eval><![CDATA[

```

```

60      ($SYS_USER=='AUTO' ||  
61       $SYS_USER=='SEEAUTO') &&  
62      ($SYS_TELRUN_STATUS=='GOOD' ||  
63       $SYS_TELRUN_STATUS=='WARNING')  
64    ]]></eval>  
65    <!-- When desired, the number of pending observations define the GOAL -->  
66    <choose>  
67      <when>  
68        <eval><!CDATA[  
69          $HK_TELRUN_NOBS_STATUS==PENDING,  
70        ]]></eval>  
71        <!-- Use a trick to add SEE when needed -->  
72        <php>substr($SYS_USER,0,strlen($SYS_USER)-4)."SCIENCE"</php>  
73        <send>update SYS_GOAL=$STDOUT</send>  
74        <log>$LOG_NO Setting SYS_GOAL=$STDOUT</log>  
75      </when>  
76      <when>  
77        <eval><!CDATA[  
78          $HK_TELRUN_NCAL_STATUS==PENDING,  
79        ]]></eval>  
80        <!-- Use a trick to add SEE when needed -->  
81        <php>substr($SYS_USER,0,strlen($SYS_USER)-4)."CALIB"</php>  
82        <send>update SYS_GOAL=$STDOUT</send>  
83        <log>$LOG_NO Setting SYS_GOAL=$STDOUT</log>  
84      </when>  
85      <when>  
86        <eval><!CDATA[  
87          $HK_TELRUN_NLINES_STATUS==EMPTY,  
88        ]]></eval>  
89        <send>update SYS_GOAL=STOP</send>  
90        <log>$LOG_NO Setting SYS_GOAL=STOP</log>  
91      </when>  
92    </choose>  
93  </when>  
94  <otherwise>  
95    <log>$LOG_N1 Invalid value for SYS_USER=$SYS_USER</log>  
96  </otherwise>  
97 </choose>  
98 </scheduler>  
99 <!-- Scheduler: STATUS. Show the current status of the system-->  
100 <scheduler>  
101   <name>status</name>  
102   <hash>  
103     <variable>HK_DOME_CIRCUITS_POWER_GOAL</variable>  
104     <variable>HK_DOME_CIRCUITS_POWER_STATUS</variable>  
105     <variable>HK_TELRUN_DAEMON_GOAL</variable>  
106     <variable>HK_TELRUN_DAEMON_STATUS</variable>  
107     <variable>HK_TJO_80V_POWER_GOAL</variable>  
108     <variable>HK_TJO_80V_POWER_STATUS</variable>  
109     <variable>SYS_GOAL</variable>  
110     <variable>SYS_DOME_STATUS</variable>  
111     <variable>SYS_MEIA_STATUS</variable>  
112     <variable>SYS_METEO_STATUS</variable>  
113     <variable>SYS_STATUS</variable>  
114     <variable>SYS_TELRUN_STATUS</variable>  
115     <variable>SYS_TJO_STATUS</variable>  
116   </hash>  
117   <log>$LOG_NO Entering STATUS scheduler</log>  
118   <php>$SYS_STATUS</php>  
119   <!-- First, check the status of the schedulers -->  
120   <choose>  
121     <when>  
122       <eval><!CDATA[  
123         $SYS_DOME_STATUS=='ERROR' ||  
124         $SYS_TJO_STATUS=='ERROR' ||  
125         $SYS_MEIA_STATUS=='ERROR' ||  
126         $SYS_TELRUN_STATUS=='ERROR',  
127       ]]></eval>  
128       <send>update SYS_STATUS=MINOR_ERROR</send>  
129     </when>  
130     <when>  
131       <eval><!CDATA[  
132         ($SYS_METEO_STATUS=='GOOD' ||  
133          $SYS_METEO_STATUS=='WARNING') &&  
134          ($SYS_DOME_STATUS=='GOOD' ||  
135            $SYS_DOME_STATUS=='WARNING') &&  
136          ($SYS_TJO_STATUS=='GOOD' ||  
137            $SYS_TJO_STATUS=='WARNING') &&  
138          ($SYS_MEIA_STATUS=='GOOD' ||  
139            $SYS_MEIA_STATUS=='WARNING') &&  
140          ($SYS_TELRUN_STATUS=='GOOD' ||  
141            $SYS_TELRUN_STATUS=='WARNING')  
142       ]]></eval>

```

```

143      <!-- Second, check the GOAL -->
144      <choose>
145          <when>
146              <eval><! [CDATA[
147                  $SYS_GOAL=='START' || 
148                  $SYS_GOAL=='INIT' || 
149                  $SYS_GOAL=='THERMAL',
150             ]]></eval>
151          <!-- Define the status according to detailed system variables -->
152          <choose>
153              <when>
154                  <eval><! [CDATA [
155                      $HK_TJO_80V_POWER_GOAL=='ON' &&
156                      $HK_TJO_80V_POWER_STATUS=='ON',
157                  ]]></eval>
158                  <send>update SYS_STATUS=$SYS_GOAL</send>
159              </when>
160              <when>
161                  <eval><! [CDATA [
162                      substr($STDOUT,-5)=='ERROR' || 
163                      $STDOUT=='RECOVERING',
164                  ]]></eval>
165                  <send>update SYS_STATUS=RECOVERING</send>
166              </when>
167              <otherwise>
168                  <send>update SYS_STATUS=STARTING</send>
169              </otherwise>
170          </choose>
171      </when>
172      <when>
173          <eval><! [CDATA [
174              substr($SYS_GOAL,-5)=='CALIB' || 
175              substr($SYS_GOAL,-7)=='SCIENCE',
176          ]]></eval>
177          <choose>
178              <when>
179                  <eval><! [CDATA [
180                      $HK_TELRUN_DAEMON_GOAL=='RUN' &&
181                      $HK_TELRUN_DAEMON_STATUS=='RUN',
182                  ]]></eval>
183                  <send>update SYS_STATUS=$SYS_GOAL</send>
184              </when>
185              <when>
186                  <eval><! [CDATA [
187                      substr($STDOUT,-5)=='ERROR' || 
188                      $STDOUT=='RECOVERING',
189                  ]]></eval>
190                  <send>update SYS_STATUS=RECOVERING</send>
191              </when>
192              <otherwise>
193                  <send>update SYS_STATUS=STARTING</send>
194              </otherwise>
195          </choose>
196      </when>
197      <when>
198          <eval><! [CDATA [
199              $SYS_GOAL=='STOP',
200          ]]></eval>
201          <!-- Define the status according to detailed system variables -->
202          <choose>
203              <when>
204                  <eval><! [CDATA [
205                      $HK_DOME_CIRCUITS_POWER_GOAL=='OFF' &&
206                      $HK_DOME_CIRCUITS_POWER_STATUS=='OFF',
207                  ]]></eval>
208                  <send>update SYS_STATUS=$SYS_GOAL</send>
209              </when>
210              <when>
211                  <eval><! [CDATA [
212                      substr($STDOUT,-5)=='ERROR' || 
213                      $STDOUT=='RECOVERING',
214                  ]]></eval>
215                  <send>update SYS_STATUS=RECOVERING</send>
216              </when>
217              <otherwise>
218                  <send>update SYS_STATUS=STOPPING</send>
219              </otherwise>
220          </choose>
221      </when>
222      <otherwise>
223          <send>update SYS_STATUS=UNKNOWN</send>
224      </otherwise>
225  </choose>

```

```

226          </when>
227          <otherwise>
228              <send>update SYS_STATUS=UNKNOWN</send>
229          </otherwise>
230      </choose>
231      <!-- Log when STATUS changes -->
232      <choose>
233          <when>
234              <eval><![CDATA[
235                  $SYS_STATUS != $STDOUT
236              ]]></eval>
237              <!-- Define the logging level -->
238              <choose>
239                  <when>
240                      <eval><![CDATA[
241                          $SYS_STATUS == MINOR_ERROR,
242                      ]]></eval>
243                      <log>$LOG_N2 Setting SYS_STATUS=MINOR_ERROR</log>
244                  </when>
245                  <when>
246                      <eval><![CDATA[
247                          $SYS_STATUS == $SYS_GOAL ||
248                          $SYS_STATUS == 'STARTING' ||
249                          $SYS_STATUS == 'STOPPING',
250                      ]]></eval>
251                      <log>$LOG_NO Setting SYS_STATUS=$SYS_STATUS</log>
252                  </when>
253                  <otherwise>
254                      <log>$LOG_N1 Setting SYS_STATUS=$SYS_STATUS</log>
255                  </otherwise>
256              </choose>
257          </when>
258      </choose>
259  </scheduler>
260  <!-- Scheduler: METEO. Actuate on meteorological systems -->
261  <scheduler>
262      <name>meteo</name>
263      <hash>
264          <variable>HK_METEO_GPSD_GOAL</variable>
265          <variable>HK_METEO_GPSD_STATUS</variable>
266          <variable>HK_METEO_WXD_READ_GOAL</variable>
267          <variable>HK_METEO_WXD_READ_STATUS</variable>
268          <variable>HK_METEO_WXD_WRITE_GOAL</variable>
269          <variable>HK_METEO_WXD_WRITE_STATUS</variable>
270          <variable>HK_TELRUN_DAEMON_STATUS</variable>
271          <variable>HK_TELRUN_DAEMON_GOAL</variable>
272          <variable>HK_TJO_80V_POWER_GOAL</variable>
273          <variable>HK_TJO_80V_POWER_STATUS</variable>
274          <variable>SYS_DOME_STATUS</variable>
275          <variable>SYS_GOAL</variable>
276          <variable>SYS_TJO_STATUS</variable>
277          <variable>SYS_METEO_STATUS</variable>
278      </hash>
279      <log>$LOG_NO Entering METEO scheduler</log>
280      <!-- Update METEO status before actuating on meteorological systems -->
281      <action>meteo.xml[status]</action>
282      <!-- Actuate on meteorological daemons -->
283      <choose>
284          <!-- Start daemons -->
285          <when>
286              <eval><![CDATA[
287                  ($SYS_GOAL == 'START' ||
288                  $SYS_GOAL == 'INIT' ||
289                  $SYS_GOAL == 'THERMAL' ||
290                  substr($SYS_GOAL, -5) == 'CALIB' ||
291                  substr($SYS_GOAL, -7) == 'SCIENCE') &&
292                  $HK_TJO_80V_POWER_GOAL == 'ON' &&
293                  $HK_TJO_80V_POWER_STATUS == 'ON' &&
294                  ($SYS_DOME_STATUS == 'GOOD' ||
295                  $SYS_DOME_STATUS == 'WARNING') &&
296                  ($SYS_TJO_STATUS == 'GOOD' ||
297                  $SYS_TJO_STATUS == 'WARNING') &&
298                  ($SYS_METEO_STATUS == 'GOOD' ||
299                  $SYS_METEO_STATUS == 'WARNING')
300              ]]></eval>
301      <choose>
302          <!-- Step 1: Start wxd.php -->
303          <when>
304              <eval><![CDATA[
305                  $HK_METEO_WXD_WRITE_GOAL == 'STOP' ||
306                  $HK_METEO_WXD_WRITE_STATUS == 'STOP'
307              ]]></eval>
308              <send>update HK_METEO_WXD_WRITE_STATUS=STARTING</send>

```

```

309          <shell>rund wxd.php -f $FS_FILE_WXD -d $TIME_NOMINAL -h $IP_METEO -e ←
310          0</shell>
311          <timeout>$TIME_SHORT</timeout>
312          <action>meteo.xml[wxd_write_status]</action>
313          <send>update HK_METEO_WXD_WRITE_GOAL=RUN</send>
314          <log>$LOG_NO Setting HK_METEO_WXD_WRITE_GOAL=RUN</log>
315      </when>
316      <!-- Step 2: Start wxd -->
317      <when>
318          <eval><!CDATA[
319              $HK_METEO_WXD_READ_GOAL=='STOP' ||
320              $HK_METEO_WXD_READ_STATUS=='STOP'
321          ]]></eval>
322          <send>update HK_METEO_WXD_READ_STATUS=STARTING</send>
323          <shell>rund -st $FS_FILE_WXD -l -</shell>
324          <timeout>$TIME_SHORT</timeout>
325          <action>meteo.xml[wxd_read_status]</action>
326          <send>update HK_METEO_WXD_READ_GOAL=RUN</send>
327          <log>$LOG_NO Setting HK_METEO_WXD_READ_GOAL=RUN</log>
328      </when>
329      <!-- Step 3: Start gpsd -->
330      <when>
331          <eval><!CDATA[
332              $HK_METEO_GPSD_GOAL=='STOP' ||
333              $HK_METEO_GPSD_STATUS=='STOP'
334          ]]></eval>
335          <send>update HK_METEO_GPSD_STATUS=STARTING</send>
336          <shell>rund gpsd -fsac</shell>
337          <timeout>$TIME_SHORT</timeout>
338          <action>meteo.xml[gpsd_status]</action>
339          <send>update HK_METEO_GPSD_GOAL=RUN</send>
340          <log>$LOG_NO Setting HK_METEO_GPSD_GOAL=RUN</log>
341      </when>
342      </choose>
343      <!-- Stop daemons -->
344      <when>
345          <eval><!CDATA[
346              ($SYS_GOAL=='STOP' ||
347              $SYS_DOME_STATUS=='ERROR' ||
348              $SYS_TJO_STATUS=='ERROR' ||
349              $SYS_METEO_STATUS=='ERROR') &&
350              ($HK_TELRUN_DAEMON_STATUS=='STOP' &&
351              $HK_TELRUN_DAEMON_GOAL=='STOP')
352          ]]></eval>
353      <choose>
354          <!-- Step 1: Stop gpsd -->
355          <when>
356              <eval><!CDATA[
357                  $HK_METEO_GPSD_STATUS != 'STOP' ||
358                  $HK_METEO_GPSD_GOAL != 'STOP'
359              ]]></eval>
360              <send>update HK_METEO_GPSD_STATUS=STOPPING</send>
361              <shell>ps ax -o pid:1,command | grep "gpsd -fsac$$" | cut -f 1,1 -d "←
362                  "</shell>
363              <timeout>$TIME_SHORT</timeout>
364              <!-- Use a trick to define the kill signal when GOAL is stop already ←
365                  -->
366              <php>15-6*strpos($HK_METEO_GPSD_GOAL,'STOP')." ".$STDOUT</php>
367              <shell>kill -$STDOUT</shell>
368              <timeout>$TIME_SHORT</timeout>
369              <shell>rm -f $FS_PID_RUND_GPSD $FS_PID_GPSD</shell>
370              <timeout>$TIME_SHORT</timeout>
371              <action>meteo.xml[gpsd_status]</action>
372              <send>update HK_METEO_GPSD_GOAL=STOP</send>
373              <log>$LOG_NO Setting HK_METEO_GPSD_GOAL=STOP</log>
374          </when>
375          <!-- Step 2: Stop wxd -->
376          <when>
377              <eval><!CDATA[
378                  $HK_METEO_WXD_READ_STATUS != 'STOP' ||
379                  $HK_METEO_WXD_READ_GOAL != 'STOP'
380              ]]></eval>
381              <send>update HK_METEO_WXD_READ_STATUS=STOPPING</send>
382              <shell>ps x -o pid:1,command | grep "wxd .* -$$" | cut -f 1,1 -d "←
383                  "</shell>
384              <timeout>$TIME_SHORT</timeout>
385              <!-- Use a trick to define the kill signal when GOAL is stop already ←
386                  -->
387              <php>15-6*strpos($HK_METEO_WXD_READ_GOAL,'STOP')." ".$STDOUT</php>
388              <shell>kill -$STDOUT</shell>
389              <timeout>$TIME_SHORT</timeout>
390              <shell>rm -f $FS_PID_RUND_WXD_READ $FS_PID_WXD_READ</shell>

```

```

387          <timeout>$TIME_SHORT</timeout>
388          <action>meteo.xml[wxd_read_status]</action>
389          <send>update HK_METEO_WXD_READ_GOAL=STOP</send>
390          <log>$LOG_NO Setting HK_METEO_WXD_READ_GOAL=STOP</log>
391      </when>
392      <!-- Step 3: Stop wxd.php -->
393      <when>
394          <eval><![CDATA[
395              $HK_METEO_WXD_WRITE_STATUS != 'STOP' ||
396              $HK_METEO_WXD_WRITE_GOAL != 'STOP'
397          ]]></eval>
398          <send>update HK_METEO_WXD_WRITE_STATUS=STOPPING</send>
399          <shell>ps x -o pid:1,command | grep "wxd.php .* -e 0$$" | cut -f 1,1 <-
400              -d " "</shell>
401          <timeout>$TIME_SHORT</timeout>
402          <!-- Use a trick to define the kill signal when GOAL is stop already -->
403          <php>15-6*strpos($HK_METEO_WXD_WRITE_GOAL , 'STOP') . " ". $STDOUT</php>
404          <shell>kill -$STDOUT</shell>
405          <timeout>$TIME_SHORT</timeout>
406          <shell>rm -f $FS_PID_RUND_WXD_WRITE</shell>
407          <timeout>$TIME_SHORT</timeout>
408          <action>meteo.xml[wxd_write_status]</action>
409          <send>update HK_METEO_WXD_WRITE_GOAL=STOP</send>
410          <log>$LOG_NO Setting HK_METEO_WXD_WRITE_GOAL=STOP</log>
411      </when>
412      </choose>
413  </choose>
414  <!-- Update METEO status before leaving -->
415  <action>meteo.xml[status]</action>
416 </scheduler>
417 <!-- Scheduler: DOME. Actuate on the dome components -->
418 <scheduler>
419     <name>dome</name>
420     <hash>
421         <variable>HK_DOME_CIRCUITS_POWER_GOAL</variable>
422         <variable>HK_DOME_CIRCUITS_POWER_STATUS</variable>
423         <variable>HK_DOME_LIGHT_POWER_GOAL</variable>
424         <variable>HK_DOME_LIGHT_POWER_STATUS</variable>
425         <variable>HK_TJO_5V_POWER_GOAL</variable>
426         <variable>HK_TJO_5V_POWER_STATUS</variable>
427         <variable>HK_TJO_80V_POWER_GOAL</variable>
428         <variable>HK_TJO_80V_POWER_STATUS</variable>
429         <variable>SYS_DOME_STATUS</variable>
430         <variable>SYS_GOAL</variable>
431         <variable>SYS_STATUS</variable>
432     </hash>
433     <log>$LOG_NO Entering DOME scheduler</log>
434     <!-- Update dome status before actuating on dome systems -->
435     <action>dome.xml[status]</action>
436     <!-- Actuate on the dome -->
437     <choose>
438         <!-- Switch on the dome circuits -->
439         <when>
440             <eval><![CDATA[
441                 ($SYS_GOAL == 'START' || 
442                  $SYS_GOAL == 'INIT' || 
443                  $SYS_GOAL == 'THERMAL' || 
444                  substr($SYS_GOAL,-5) == 'CALIB' || 
445                  substr($SYS_GOAL,-7) == 'SCIENCE') &&
446                 ($SYS_DOME_STATUS == 'GOOD' || 
447                  $SYS_DOME_STATUS == 'WARNING') &&
448                 ($HK_DOME_CIRCUITS_POWER_GOAL == 'OFF' || 
449                  $HK_DOME_CIRCUITS_POWER_STATUS == 'OFF')
450             ]]></eval>
451             <send>update HK_DOME_CIRCUITS_POWER_STATUS=STARTING</send>
452             <shell>snmpset -v 1 -c private $IP_PDU_NO $OID_PDU_N6 i $PDU_ON</shell>
453             <action>dome.xml[circuits_pdu_status]</action>
454             <send>update HK_DOME_CIRCUITS_POWER_GOAL=ON</send>
455             <log>$LOG_NO Setting HK_DOME_CIRCUITS_POWER_GOAL=ON</log>
456         </when>
457         <!-- Switch off the dome circuits -->
458         <when>
459             <eval><![CDATA[
460                 ($SYS_GOAL == 'STOP' || 
461                  $SYS_DOME_STATUS == 'ERROR') &&
462                 ($HK_TJO_5V_POWER_GOAL == 'OFF' &&
463                  $HK_TJO_5V_POWER_STATUS == 'OFF') &&
464                 ($HK_DOME_CIRCUITS_POWER_STATUS != 'OFF' || 
465                  $HK_DOME_CIRCUITS_POWER_GOAL != 'OFF')
466             ]]></eval>
467             <send>update HK_DOME_CIRCUITS_POWER_STATUS=STOPPING</send>

```

```

468      <shell>snmpset -v 1 -c private $IP_PDU_NO $OID_PDU_N6 i $PDU_OFF</shell>
469      <action>dome.xml[circuits_pdu_status]</action>
470      <send>update HK_DOME_CIRCUITS_POWER_GOAL=OFF</send>
471      <log>$LOG_NO Setting HK_DOME_CIRCUITS_POWER_GOAL=OFF</log>
472      </when>
473  </choose>
474  <!-- Update dome status -->
475  <action>dome.xml[status]</action>
476  <!-- Actuate on the dome light -->
477  <choose>
478      <!-- Switch on light -->
479  <when>
480      <eval><![CDATA[
481          ($SYS_GOAL=='START' || 
482           $SYS_GOAL=='INIT' || 
483           $SYS_GOAL=='THERMAL' || 
484           substr($SYS_GOAL,0,3)=='SEE' || 
485           substr($SYS_STATUS,-5)=='ERROR') &&
486           ($HK_DOME_LIGHT_POWER_GOAL=='OFF' || 
487            $HK_DOME_LIGHT_POWER_STATUS=='OFF')
488      ]]></eval>
489      <send>update HK_DOME_LIGHT_POWER_STATUS=STARTING</send>
490  <shell>snmpset -v 1 -c private $IP_PDU_NO $OID_PDU_N7 i $PDU_ON</shell>
491  <action>dome.xml[light_pdu_status]</action>
492  <send>update HK_DOME_LIGHT_POWER_GOAL=ON</send>
493  <log>$LOG_NO Setting HK_DOME_LIGHT_POWER_GOAL=ON</log>
494  </when>
495  <!-- Switch off light -->
496  <when>
497      <eval><![CDATA[
498          (($SYS_GOAL=='CALIB' || 
499             $SYS_GOAL=='SCIENCE') &&
500             ($HK_TJO_80V_POWER_GOAL=='ON' &&
501              $HK_TJO_80V_POWER_STATUS=='ON') || 
502              ($SYS_GOAL=='STOP' &&
503                 $HK_DOME_CIRCUITS_POWER_GOAL=='OFF' &&
504                 $HK_DOME_CIRCUITS_POWER_STATUS=='OFF')) &&
505                 substr($SYS_STATUS,-5)!='ERROR' &&
506                 ($HK_DOME_LIGHT_POWER_GOAL!='OFF' || 
507                  $HK_DOME_LIGHT_POWER_STATUS!='OFF')
508      ]]></eval>
509      <send>update HK_DOME_LIGHT_POWER_STATUS=STOPPING</send>
510  <shell>snmpset -v 1 -c private $IP_PDU_NO $OID_PDU_N7 i $PDU_OFF</shell>
511  <action>dome.xml[light_pdu_status]</action>
512  <send>update HK_DOME_LIGHT_POWER_GOAL=OFF</send>
513  <log>$LOG_NO Setting HK_DOME_LIGHT_POWER_GOAL=OFF</log>
514  </when>
515  </choose>
516  <!-- Update dome status before leaving -->
517  <action>dome.xml[status]</action>
518 </scheduler>
519  <!-- Scheduler: TJO. Actuate on the TJO components -->
520 <scheduler>
521  <name>tjo</name>
522  <hash>
523      <variable>HK_DOME_CIRCUITS_POWER_GOAL</variable>
524      <variable>HK_DOME_CIRCUITS_POWER_STATUS</variable>
525      <variable>HK_TJO_5V_POWER_GOAL</variable>
526      <variable>HK_TJO_5V_POWER_STATUS</variable>
527      <variable>HK_TJO_12V_POWER_GOAL</variable>
528      <variable>HK_TJO_12V_POWER_STATUS</variable>
529      <variable>HK_TJO_24V_POWER_GOAL</variable>
530      <variable>HK_TJO_24V_POWER_STATUS</variable>
531      <variable>HK_TJO_80V_POWER_GOAL</variable>
532      <variable>HK_TJO_80V_POWER_STATUS</variable>
533      <variable>HK_TJO_CSIMCD_GOAL</variable>
534      <variable>HK_TJO_CSIMCD_STATUS</variable>
535      <variable>HK_TJO_TELESCOPED_GOAL</variable>
536      <variable>HK_TJO_TELESCOPED_STATUS</variable>
537      <variable>HK_TELRUN_DAEMON_STATUS</variable>
538      <variable>HK_TELRUN_DAEMON_GOAL</variable>
539      <variable>SYS_DOME_STATUS</variable>
540      <variable>SYS_GOAL</variable>
541      <variable>SYS_TJO_STATUS</variable>
542  </hash>
543  <log>$LOG_NO Entering TJO scheduler</log>
544  <!-- Update TJO status before actuating on TJO systems -->
545  <action>tjo.xml[status]</action>
546  <!-- Actuate on the TJO -->
547  <choose>
548      <!-- Start TJO -->
549  <when>
550      <eval><![CDATA[
```

```

551      ($SYS_GOAL== 'START' ||  

552      $SYS_GOAL== 'INIT' ||  

553      $SYS_GOAL== 'THERMAL' ||  

554      substr($SYS_GOAL,-5)== 'CALIB' ||  

555      substr($SYS_GOAL,-7)== 'SCIENCE') &&  

556      $HK_DOME_CIRCUITS_POWER_GOAL== 'ON' &&  

557      $HK_DOME_CIRCUITS_POWER_STATUS== 'ON' &&  

558      ($SYS_DOME_STATUS== 'GOOD' ||  

559      $SYS_DOME_STATUS== 'WARNING') &&  

560      ($SYS_TJO_STATUS== 'GOOD' ||  

561      $SYS_TJO_STATUS== 'WARNING')  

562  ]]></eval>  

563  <choose>  

564    <!-- Step 1: Switch on 5V power -->  

565    <when>  

566      <eval><! [CDATA[  

567        $HK_TJO_5V_POWER_GOAL== 'OFF' ||  

568        $HK_TJO_5V_POWER_STATUS== 'OFF'  

569      ]]></eval>  

570      <send>update HK_TJO_5V_POWER_STATUS=STARTING</send>  

571      <shell>snmpset -v 1 -c private $IP_PDU_N1 $OID_PDU_N1 i $PDU_ON</-->  

572      shell>  

573      <action>tjo.xml [5v_pdu_status]</action>  

574      <send>update HK_TJO_5V_POWER_GOAL=ON</send>  

575      <log>$LOG_NO Setting HK_TJO_5V_POWER_GOAL=ON</log>  

576    </when>  

577    <!-- Step 2: Switch on 12V power -->  

578    <when>  

579      <eval><! [CDATA[  

580        $HK_TJO_12V_POWER_GOAL== 'OFF' ||  

581        $HK_TJO_12V_POWER_STATUS== 'OFF'  

582      ]]></eval>  

583      <send>update HK_TJO_12V_POWER_STATUS=STARTING</send>  

584      <shell>snmpset -v 1 -c private $IP_PDU_N1 $OID_PDU_N4 i $PDU_ON</-->  

585      shell>  

586      <action>tjo.xml [12v_pdu_status]</action>  

587      <send>update HK_TJO_12V_POWER_GOAL=ON</send>  

588      <log>$LOG_NO Setting HK_TJO_12V_POWER_GOAL=ON</log>  

589    </when>  

590    <!-- Step 3: Switch on 24V power -->  

591    <when>  

592      <eval><! [CDATA[  

593        $HK_TJO_24V_POWER_GOAL== 'OFF' ||  

594        $HK_TJO_24V_POWER_STATUS== 'OFF'  

595      ]]></eval>  

596      <send>update HK_TJO_24V_POWER_STATUS=STARTING</send>  

597      <shell>snmpset -v 1 -c private $IP_PDU_N1 $OID_PDU_N2 i $PDU_ON</-->  

598      shell>  

599      <action>tjo.xml [24v_pdu_status]</action>  

600      <send>update HK_TJO_24V_POWER_GOAL=ON</send>  

601      <log>$LOG_NO Setting HK_TJO_24V_POWER_GOAL=ON</log>  

602    </when>  

603    <!-- Step 4: Start telescopeds+csimcd -->  

604    <when>  

605      <eval><! [CDATA[  

606        $HK_TJO_CSIMCD_GOAL== 'STOP' ||  

607        $HK_TJO_CSIMCD_STATUS== 'STOP' ||  

608        $HK_TJO_TELESCOPED_GOAL== 'STOP' ||  

609        $HK_TJO_TELESCOPED_STATUS== 'STOP'  

610      ]]></eval>  

611      <send>update HK_TJO_TELESCOPED_STATUS=STARTING</send>  

612      <send>update HK_TJO_CSIMCD_STATUS=STARTING</send>  

613      <shell>rund telescopeds</shell>  

614      <timeout>$TIME_SHORT</timeout>  

615      <shell>tail -n 0 -f $FS_LOG_TELESCOPED | grep "cover.cfg" -m 1</shell>  

616      <timeout>$TIME_RUND_TELESCOPED</timeout>  

617      <action>tjo.xml [telescopeds_status]</action>  

618      <send>update HK_TJO_TELESCOPED_GOAL=RUN</send>  

619      <log>$LOG_NO Setting HK_TJO_TELESCOPED_GOAL=RUN</log>  

620    </when>  

621    <!-- Step 5: Switch on 80V power -->  

622    <when>  

623      <eval><! [CDATA[  

624        $HK_TJO_80V_POWER_GOAL== 'OFF' ||  

625        $HK_TJO_80V_POWER_STATUS== 'OFF'  

626      ]]></eval>  

627      <send>update HK_TJO_80V_POWER_STATUS=STARTING</send>  

628      <shell>snmpset -v 1 -c private $IP_PDU_N1 $OID_PDU_N3 i $PDU_ON</-->  

       shell>

```

```

629          <action>tjo.xml[80v_pdu_status]</action>
630          <send>update HK_TJO_80V_POWER_GOAL=ON</send>
631          <log>$LOG_NO Setting HK_TJO_80V_POWER_GOAL=ON</log>
632      </when>
633  </choose>
634  <!-- Stop the TJO -->
635  <when>
636      <eval><![CDATA[
637          ($SYS_GOAL=='STOP' || 
638             $SYS_DOME_STATUS=='ERROR' || 
639             $SYS_TJO_STATUS=='ERROR') &&
640             $HK_TELRUN_DAEMON_STATUS=='STOP' &&
641             $HK_TELRUN_DAEMON_GOAL=='STOP',
642         ]]></eval>
643  <choose>
644      <!-- Step 1: Switch off 80V power -->
645      <when>
646          <eval><![CDATA[
647              $HK_TJO_80V_POWER_STATUS != 'OFF' || 
648              $HK_TJO_80V_POWER_GOAL != 'OFF'
649          ]]></eval>
650          <send>update HK_TJO_80V_POWER_STATUS=STOPPING</send>
651          <shell>snmpset -v 1 -c private $IP_PDU_N1 $OID_PDU_N3 i $PDU_OFF</->
652          <shell>
653          <action>tjo.xml[80v_pdu_status]</action>
654          <send>update HK_TJO_80V_POWER_GOAL=OFF</send>
655          <log>$LOG_NO Setting HK_TJO_80V_POWER_GOAL=OFF</log>
656      </when>
657      <!-- Step 2: Stop telescoped -->
658      <when>
659          <eval><![CDATA[
660              $HK_TJO_TELESCOPED_STATUS != 'STOP' || 
661              $HK_TJO_TELESCOPED_GOAL != 'STOP'
662          ]]></eval>
663          <send>update HK_TJO_TELESCOPED_STATUS=STOPPING</send>
664          <shell>ps x -o pid:1,command | grep "telescoped$$" | cut -f 1,1 -d " " <->
665          <shell>
666          <timeout>$TIME_SHORT</timeout>
667          <!-- Use a trick to define the kill signal when GOAL is stop already -->
668          <php>15-6*strpos($HK_TJO_TELESCOPED_GOAL , 'STOP').". ".$STDOUT</php>
669          <shell>kill -$STDOUT</shell>
670          <timeout>$TIME_SHORT</timeout>
671          <shell>rm -f $FS_PID_RUND_TELESCOPED $FS_PID_TELESCOPED </shell>
672          <timeout>$TIME_SHORT</timeout>
673          <shell>rm -f $FS_FIFO_COVER_IN $FS_FIFO_COVER_OUT </shell>
674          <timeout>$TIME_SHORT</timeout>
675          <shell>rm -f $FS_FIFO_DOME_IN $FS_FIFO_DOME_OUT </shell>
676          <timeout>$TIME_SHORT</timeout>
677          <shell>rm -f $FS_FIFO_FILTER_IN $FS_FIFO_FILTER_OUT </shell>
678          <timeout>$TIME_SHORT</timeout>
679          <shell>rm -f $FS_FIFO_FOCUS_IN $FS_FIFO_FOCUS_OUT </shell>
680          <timeout>$TIME_SHORT</timeout>
681          <shell>rm -f $FS_FIFO_TEL_IN $FS_FIFO_TEL_OUT </shell>
682          <timeout>$TIME_SHORT</timeout>
683          <shell>rm -f $FS_FIFO_LIGHTS_IN $FS_FIFO_LIGHTS_OUT </shell>
684          <timeout>$TIME_SHORT</timeout>
685          <shell>rm -f $FS_FIFO_POWERFAIL_IN $FS_FIFO_POWERFAIL_OUT </shell>
686          <timeout>$TIME_SHORT</timeout>
687          <action>tjo.xml[telescoped_status]</action>
688          <send>update HK_TJO_TELESCOPED_GOAL=STOP</send>
689          <log>$LOG_NO Setting HK_TJO_TELESCOPED_GOAL=STOP</log>
690      </when>
691      <!-- Step 3: Stop csimcd -->
692      <when>
693          <eval><![CDATA[
694              $HK_TJO_CSIMCD_STATUS != 'STOP' || 
695              $HK_TJO_CSIMCD_GOAL != 'STOP'
696          ]]></eval>
697          <send>update HK_TJO_CSIMCD_STATUS=STOPPING</send>
698          <shell>ps x -o pid:1,command | grep "csimcd.*$PORT_CSIMCD$$" | cut -f<->
699          1,1 -d " "</shell>
700          <timeout>$TIME_SHORT</timeout>
701          <!-- Use a trick to define the kill signal when GOAL is stop already -->
702          <php>15-6*strpos($HK_TJO_CSIMCD_GOAL , 'STOP').". ".$STDOUT</php>
703          <shell>kill -$STDOUT</shell>
704          <timeout>$TIME_SHORT</timeout>
705          <action>tjo.xml[csimcd_status]</action>
706          <send>update HK_TJO_CSIMCD_GOAL=STOP</send>

```

```

707             <log>$LOG_NO Setting HK_TJO_CSIMCD_GOAL=STOP</log>
708         </when>
709         <!-- Step 4: Switch off 24V power -->
710         <when>
711             <eval><![CDATA[
712                 $HK_TJO_24V_POWER_STATUS != 'OFF' || 
713                 $HK_TJO_24V_POWER_GOAL != 'OFF'
714             ]]></eval>
715             <send>update HK_TJO_24V_POWER_STATUS=STOPPING</send>
716             <shell>snmpset -v 1 -c private $IP_PDU_N1 $OID_PDU_N2 i $PDU_OFF</->
717                     shell>
718             <action>tjo.xml[24v_pdu_status]</action>
719             <send>update HK_TJO_24V_POWER_GOAL=OFF</send>
720             <log>$LOG_NO Setting HK_TJO_24V_POWER_GOAL=OFF</log>
721         </when>
722         <!-- Step 5: Switch off 12V power -->
723         <when>
724             <eval><![CDATA[
725                 $HK_TJO_12V_POWER_STATUS != 'OFF' || 
726                 $HK_TJO_12V_POWER_GOAL != 'OFF'
727             ]]></eval>
728             <send>update HK_TJO_12V_POWER_STATUS=STOPPING</send>
729             <shell>snmpset -v 1 -c private $IP_PDU_N1 $OID_PDU_N4 i $PDU_OFF</->
730                     shell>
731             <action>tjo.xml[12v_pdu_status]</action>
732             <send>update HK_TJO_12V_POWER_GOAL=OFF</send>
733             <log>$LOG_NO Setting HK_TJO_12V_POWER_GOAL=OFF</log>
734         </when>
735         <!-- Step 6: Switch off 5V power -->
736         <when>
737             <eval><![CDATA[
738                 $HK_TJO_5V_POWER_STATUS != 'OFF' || 
739                 $HK_TJO_5V_POWER_GOAL != 'OFF'
740             ]]></eval>
741             <send>update HK_TJO_5V_POWER_STATUS=STOPPING</send>
742             <shell>snmpset -v 1 -c private $IP_PDU_N1 $OID_PDU_N1 i $PDU_OFF</->
743                     shell>
744             <action>tjo.xml[5v_pdu_status]</action>
745             <send>update HK_TJO_5V_POWER_GOAL=OFF</send>
746             <log>$LOG_NO Setting HK_TJO_5V_POWER_GOAL=OFF</log>
747         </when>
748     </choose>
749     <!-- Update TJO status before leaving -->
750     <action>tjo.xml[status]</action>
751   </scheduler>
752   <!-- Scheduler: MEIA. Actuate on MEIA instrument -->
753   <scheduler>
754     <name>meia</name>
755     <hash>
756       <variable>HK_MEIA_CAMERAD_GOAL</variable>
757       <variable>HK_MEIA_CAMERAD_STATUS</variable>
758       <variable>HK_MEIA_TEMP_GOAL</variable>
759       <variable>HK_MEIA_TEMP_STATUS</variable>
760       <variable>HK_TELRUN_DAEMON_STATUS</variable>
761       <variable>HK_TELRUN_DAEMON_GOAL</variable>
762       <variable>HK_TJO_80V_POWER_GOAL</variable>
763       <variable>HK_TJO_80V_POWER_STATUS</variable>
764       <variable>SYS_DOME_STATUS</variable>
765       <variable>SYS_GOAL</variable>
766       <variable>SYS_MEIA_STATUS</variable>
767       <variable>SYS_TJO_STATUS</variable>
768     </hash>
769     <log>$LOG_NO Entering MEIA scheduler</log>
770     <!-- Update MEIA status before actuating on MEIA systems -->
771     <action>meia.xml[status]</action>
772     <!-- Actuate on MEIA -->
773     <choose>
774       <!-- Start MEIA -->
775       <when>
776           <eval><![CDATA[
777               ($SYS_GOAL == 'START' || 
778               $SYS_GOAL == 'INIT' || 
779               $SYS_GOAL == 'THERMAL' || 
780               substr($SYS_GOAL, -5) == 'CALIB' || 
781               substr($SYS_GOAL, -7) == 'SCIENCE') &&
782               $HK_TJO_80V_POWER_GOAL == 'ON' &&
783               $HK_TJO_80V_POWER_STATUS == 'ON' &&
784               ($SYS_DOME_STATUS == 'GOOD' || 
785               $SYS_DOME_STATUS == 'WARNING') &&
786               ($SYS_TJO_STATUS == 'GOOD' || 
787               $SYS_TJO_STATUS == 'WARNING') &&
788           ]]></eval>
789       </when>
790     </choose>

```

```

787
788     ($SYS_MEIA_STATUS=='GOOD' ||  

789      $SYS_MEIA_STATUS=='WARNING') &&  

790      ($HK_MEIA_CAMERAD_GOAL=='STOP' ||  

791      $HK_MEIA_CAMERAD_STATUS=='STOP')  

792   ]]></eval>  

793   <send>update HK_MEIA_CAMERAD_STATUS=STARTING</send>  

794   <shell>rund camerad</shell>  

795   <timeout>$TIME_SHORT</timeout>  

796   <shell>tail -n 0 -f $FS_LOG_CAMERAD | grep "ID_INSTRUME = MEIA" -m 1</shell>  

797   <timeout>$TIME_RUND_CAMERAD</timeout>  

798   <action>meia.xml[camerad_status]</action>  

799   <send>update HK_MEIA_CAMERAD_GOAL=RUN</send>  

800   <log>$LOG_NO Setting HK_MEIA_CAMERAD_GOAL=RUN</log>  

801   <action>meia.xml[temp_status]</action>  

802   <send>update HK_MEIA_TEMP_GOAL=COOL</send>  

803   <log>$LOG_NO Setting HK_MEIA_TEMP_GOAL=COOL</log>  

804   </when>  

805   <!-- Stop MEIA -->  

806   <when>  

807     <eval><![CDATA[  

808       ($SYS_GOAL=='STOP' ||  

809        $SYS_DOME_STATUS=='ERROR' ||  

810        $SYS_TJO_STATUS=='ERROR' ||  

811        $SYS_MEIA_STATUS=='ERROR') &&  

812        ($HK_TELRUN_DAEMON_STATUS=='STOP' &&  

813         $HK_TELRUN_DAEMON_GOAL=='STOP')  

814     ]]></eval>  

815     <send>update HK_MEIA_CAMERAD_STATUS=STOPPING</send>  

816     <shell>ps x -o pid:1,command | grep "camerad$$" | cut -f 1,1 -d " "</shell>  

817     <timeout>$TIME_SHORT</timeout>  

818     <!-- Use a trick to define the kill signal when GOAL is stop already -->  

819     <php>15-6*strpos($HK_MEIA_CAMERAD_GOAL,'STOP')." ".$STDOUT</php>  

820     <shell>kill -$STDOUT</shell>  

821     <timeout>$TIME_SHORT</timeout>  

822     <shell>rm -f $FS_PID_RUND_CAMERAD $FS_PID_CAMERAD</shell>  

823     <timeout>$TIME_SHORT</timeout>  

824     <shell>rm -f $FS_FIFO_CAMERA_IN $FS_FIFO_CAMERA_OUT</shell>  

825     <timeout>$TIME_SHORT</timeout>  

826     <action>meia.xml[camerad_status]</action>  

827     <send>update HK_MEIA_CAMERAD_GOAL=STOP</send>  

828     <log>$LOG_NO Setting HK_MEIA_CAMERAD_GOAL=STOP</log>  

829     <action>meia.xml[temp_status]</action>  

830     <send>update HK_MEIA_TEMP_GOAL=OFF</send>  

831     <log>$LOG_NO Setting HK_MEIA_TEMP_GOAL=OFF</log>  

832   </when>  

833   </choose>  

834   <!-- Update MEIA status before leaving -->  

835   <action>meia.xml[status]</action>  

836 </scheduler>  

837   <!-- Scheduler: TELRUN. Actuate on TELRUN daemon -->  

838   <scheduler>  

839     <name>telrun</name>  

840     <hash>  

841       <variable>HK_DOME_LIGHT_POWER_STATUS</variable>  

842       <variable>HK_DOME_LIGHT_POWER_GOAL</variable>  

843       <variable>HK_TELRUN_DAEMON_STATUS</variable>  

844       <variable>HK_TELRUN_DAEMON_GOAL</variable>  

845       <variable>HK_TELRUN_NCAL_STATUS</variable>  

846       <variable>HK_TELRUN_NLINES_STATUS</variable>  

847       <variable>HK_TELRUN_NOBS_STATUS</variable>  

848       <variable>HK_TJO_80V_POWER_GOAL</variable>  

849       <variable>HK_TJO_80V_POWER_STATUS</variable>  

850       <variable>SYS_DOME_STATUS</variable>  

851       <variable>SYS_GOAL</variable>  

852       <variable>SYS_MEIA_STATUS</variable>  

853       <variable>SYS_TJO_STATUS</variable>  

854     </hash>  

855     <log>$LOG_NO Entering TELRUN scheduler</log>  

856     <!-- Update TELRUN status before actuating on telrun -->  

857     <action>telrun.xml[status]</action>  

858     <!-- Actuate on telrun daemon -->  

859     <choose>  

860       <!-- Start telrun -->  

861       <when>  

862         <eval><![CDATA[  

863           (((($SYS_GOAL=='CALIB' ||  

864             $SYS_GOAL=='SCIENCE') &&  

865             ($HK_DOME_LIGHT_POWER_STATUS=='OFF' &&  

866              $HK_DOME_LIGHT_POWER_GOAL=='OFF')) ||  

867             (((($SYS_GOAL=='SEECALIB' ||  

868               $SYS_GOAL=='SEESCIENCE') &&  

869               ($HK_DOME_LIGHT_POWER_STATUS=='ON' &&  

870                 $HK_DOME_LIGHT_POWER_GOAL=='ON')))) &&
```

```

780     $HK_TJO_80V_POWER_GOAL== 'ON' &&
781     $HK_TJO_80V_POWER_STATUS== 'ON' &&
782     $SYS_DOME_STATUS== 'GOOD' &&
783     $SYS_MEIA_STATUS== 'GOOD' &&
784     ($SYS_TJO_STATUS== 'GOOD' ||
785      $SYS_TJO_STATUS== 'WARNING') &&
786     ($SYS_TELRUN_STATUS== 'GOOD' ||
787      $SYS_TELRUN_STATUS== 'WARNING') &&
788     ($HK_TELRUN_DAEMON_STATUS== 'STOP' ||
789      $HK_TELRUN_DAEMON_GOAL== 'STOP')
790   ]]></eval>
791   <send>update HK_TELRUN_DAEMON_STATUS=STARTING</send>
792   <shell>rund telrun</shell>
793   <timeout>$TIME_SHORT</timeout>
794   <action>telrun.xml[daemon_status]</action>
795   <send>update HK_TELRUN_DAEMON_GOAL=RUN</send>
796   <log>$LOG_NO Setting HK_TELRUN_DAEMON_GOAL=RUN</log>
797 </when>
798 <!-- Stop telrun -->
799 <when>
800   <eval><![CDATA[
801     ($SYS_GOAL== 'STOP' ||
802      $SYS_GOAL== 'START' ||
803      $SYS_GOAL== 'INIT' ||
804      $SYS_GOAL== 'THERMAL' ||
805      $SYS_DOME_STATUS== 'ERROR' ||
806      $SYS_DOME_STATUS== 'WARNING' ||
807      $SYS_TJO_STATUS== 'ERROR' ||
808      $SYS_MEIA_STATUS== 'ERROR' ||
809      $SYS_MEIA_STATUS== 'WARNING' ||
810      $SYS_TELRUN_STATUS== 'ERROR') &&
811     ($HK_DOME_LIGHT_POWER_STATUS!= 'OFF' ||
812      $HK_DOME_LIGHT_POWER_GOAL!= 'OFF') &&
813     ($HK_TELRUN_DAEMON_STATUS!= 'STOP' ||
814      $HK_TELRUN_DAEMON_GOAL!= 'STOP')
815   ]]></eval>
816   <send>update HK_TELRUN_DAEMON_STATUS=STOPPING</send>
817   <shell>ps x -o pid:1,command | grep "telrun$$" | cut -f 1,1 -d " "</shell>
818   <timeout>$TIME_SHORT</timeout>
819   <!-- Use a trick to define the kill signal when GOAL is stop already -->
820   <php>15-6*strpos($HK_TELRUN_DAEMON_GOAL,'STOP').". ".$STDOUT</php>
821   <shell>kill -$STDOUT</shell>
822   <timeout>$TIME_SHORT</timeout>
823   <shell>rm -f $FS_PID_RUND_TELRUN $FS_PID_TELRUN</shell>
824   <timeout>$TIME_SHORT</timeout>
825   <action>telrun.xml[daemon_status]</action>
826   <send>update HK_TELRUN_DAEMON_GOAL=STOP</send>
827   <log>$LOG_NO Setting HK_TELRUN_DAEMON_GOAL=STOP</log>
828 </when>
829 </choose>
830 <!-- Update TELRUN status before leaving -->
831 <action>telrun.xml[status]</action>
832 </scheduler>
833 </root>

```

12.10.5. tjo.xml

```
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <!--
3   -----
4   / _ \ - -- | ----- | _ \ / _ \ / _ \ / _ \ | _ \ / _ \ / _ \
5   | | | | , \ / _ \ / _ \ | | | | | | | | | | | | | | | | | | |
6   | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
7   \_ \_ \_ / \_ \_ \_ / \_ \_ \_ / \_ \_ \_ / \_ \_ \_ / \_ \_ \_ / \_ \_ \_ / \_ \_ \_ /
8   | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
9
10 OpenROCS: Open Robotic Observatory Control System
11 Copyright (C) 2011-2014 by Institut d'Estudis Espacials de Catalunya (IEEC)
12 More information in http://www.ieec.cat or iec@ieec.cat
13
14 This program is free software: you can redistribute it and/or modify
15 it under the terms of the GNU General Public License as published by
16 the Free Software Foundation, either version 3 of the License, or
17 (at your option) any later version.
18
19 This program is distributed in the hope that it will be useful,
20 but WITHOUT ANY WARRANTY; without even the implied warranty of
21 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
22
```

```

22  GNU General Public License for more details.
23
24 You should have received a copy of the GNU General Public License
25 along with this program. If not, see <http://www.gnu.org/licenses/>.
26 -->
27 <root>
28     <!-- Actions: TJO. Update variables for TJO -->
29     <actions>
30         <!-- Check the TJO status -->
31         <action>
32             <name>status</name>
33             <php>$SYS_TJO_STATUS </php>
34             <choose>
35                 <when>
36                     <eval><! [CDATA [
37                         $HK_TJO_5V_POWER_STATUS != $HK_TJO_5V_POWER_GOAL ||
38                         $HK_TJO_12V_POWER_STATUS != $HK_TJO_12V_POWER_GOAL ||
39                         $HK_TJO_24V_POWER_STATUS != $HK_TJO_24V_POWER_GOAL ||
40                         $HK_TJO_80V_POWER_STATUS != $HK_TJO_80V_POWER_GOAL ||
41                         ($HK_TELRUN_DAEMON_STATUS == 'STOP' &&
42                             (substr($HK_TJO_CSIMCD_STATUS,-strlen($HK_TJO_CSIMCD_GOAL)) != <!--
43                                 $HK_TJO_CSIMCD_GOAL ||
44                                 substr($HK_TJO_TELESCOPED_STATUS,-strlen($HK_TJO_TELESCOPED_GOAL)) <!--
45                                 != $HK_TJO_TELESCOPED_GOAL))
46                         ]]></eval>
47                         <send>update SYS_TJO_STATUS=ERROR </send>
48                 </when>
49                 <when>
50                     <eval><! [CDATA [
51                         $HK_TELRUN_DAEMON_STATUS == 'RUN' &&
52                             ($HK_TJO_CSIMCD_GOAL != $HK_TJO_CSIMCD_STATUS ||
53                             $HK_TJO_TELESCOPED_GOAL != $HK_TJO_TELESCOPED_STATUS)
54                         ]]></eval>
55                         <send>update SYS_TJO_STATUS=WARNING </send>
56                 </when>
57                 <when>
58                     <eval><! [CDATA [
59                         $HK_TJO_5V_POWER_STATUS == $HK_TJO_5V_POWER_GOAL &&
60                         $HK_TJO_12V_POWER_STATUS == $HK_TJO_12V_POWER_GOAL &&
61                         $HK_TJO_24V_POWER_STATUS == $HK_TJO_24V_POWER_GOAL &&
62                         $HK_TJO_80V_POWER_STATUS == $HK_TJO_80V_POWER_GOAL &&
63                         $HK_TJO_CSIMCD_GOAL == $HK_TJO_CSIMCD_STATUS &&
64                         $HK_TJO_TELESCOPED_GOAL == $HK_TJO_TELESCOPED_STATUS
65                         ]]></eval>
66                         <send>update SYS_TJO_STATUS=GOOD </send>
67                 </when>
68                 <otherwise>
69                     <send>update SYS_TJO_STATUS=UNKNOWN </send>
70                 </otherwise>
71             <choose>
72                 <when>
73                     <eval><! [CDATA [
74                         $SYS_TJO_STATUS != $STDOUT
75                         ]]></eval>
76                     <!-- Define the logging level -->
77                     <choose>
78                         <when>
79                             <eval><! [CDATA [
80                                 $SYS_TJO_STATUS == 'ERROR',
81                                 ]]></eval>
82                                 <log>$LOG_N2 Setting SYS_TJO_STATUS=$SYS_TJO_STATUS </log>
83                         </when>
84                         <when>
85                             <eval><! [CDATA [
86                                 $SYS_TJO_STATUS == 'WARNING',
87                                 ]]></eval>
88                                 <log>$LOG_N1 Setting SYS_TJO_STATUS=$SYS_TJO_STATUS </log>
89                         </when>
90                         <when>
91                             <eval><! [CDATA [
92                                 $SYS_TJO_STATUS == 'GOOD',
93                                 ]]></eval>
94                                 <log>$LOG_NO Setting SYS_TJO_STATUS=$SYS_TJO_STATUS </log>
95                         </when>
96                         <otherwise>
97                             <log>$LOG_N1 Setting SYS_TJO_STATUS=$SYS_TJO_STATUS </log>
98                         </otherwise>
99                     <choose>
100                         <when>
101                         </choose>
102                     </action>

```

```

103      <!-- Check TJO power supply: 5V -->
104      <action>
105          <name>5v_pdu_status</name>
106          <shell>snmpget -v1 -c public $IP_PDU_N1 $OID_PDU_N1</shell>
107          <choose>
108              <when>
109                  <eval>$STDOUT4===$PDU_ON</eval>
110                  <send>update HK_TJO_5V_POWER_STATUS=ON</send>
111              </when>
112              <when>
113                  <eval>$STDOUT4===$PDU_OFF</eval>
114                  <send>update HK_TJO_5V_POWER_STATUS=OFF</send>
115              </when>
116              <otherwise>
117                  <send>update HK_TJO_5V_POWER_STATUS=ERROR</send>
118              </otherwise>
119          </choose>
120      </action>
121      <!-- Check TJO power supply: 12V -->
122      <action>
123          <name>12v_pdu_status</name>
124          <shell>snmpget -v1 -c public $IP_PDU_N1 $OID_PDU_N4</shell>
125          <choose>
126              <when>
127                  <eval>$STDOUT4===$PDU_ON</eval>
128                  <send>update HK_TJO_12V_POWER_STATUS=ON</send>
129              </when>
130              <when>
131                  <eval>$STDOUT4===$PDU_OFF</eval>
132                  <send>update HK_TJO_12V_POWER_STATUS=OFF</send>
133              </when>
134              <otherwise>
135                  <send>update HK_TJO_12V_POWER_STATUS=ERROR</send>
136              </otherwise>
137          </choose>
138      </action>
139      <!-- Check TJO power supply: 24V -->
140      <action>
141          <name>24v_pdu_status</name>
142          <shell>snmpget -v1 -c public $IP_PDU_N1 $OID_PDU_N2</shell>
143          <choose>
144              <when>
145                  <eval>$STDOUT4===$PDU_ON</eval>
146                  <send>update HK_TJO_24V_POWER_STATUS=ON</send>
147              </when>
148              <when>
149                  <eval>$STDOUT4===$PDU_OFF</eval>
150                  <send>update HK_TJO_24V_POWER_STATUS=OFF</send>
151              </when>
152              <otherwise>
153                  <send>update HK_TJO_24V_POWER_STATUS=ERROR</send>
154              </otherwise>
155          </choose>
156      </action>
157      <!-- Check TJO power supply: 80V -->
158      <action>
159          <name>80v_pdu_status</name>
160          <shell>snmpget -v1 -c public $IP_PDU_N1 $OID_PDU_N3</shell>
161          <choose>
162              <when>
163                  <eval>$STDOUT4===$PDU_ON</eval>
164                  <send>update HK_TJO_80V_POWER_STATUS=ON</send>
165              </when>
166              <when>
167                  <eval>$STDOUT4===$PDU_OFF</eval>
168                  <send>update HK_TJO_80V_POWER_STATUS=OFF</send>
169              </when>
170              <otherwise>
171                  <send>update HK_TJO_80V_POWER_STATUS=ERROR</send>
172              </otherwise>
173          </choose>
174      </action>
175      <!-- Check telescoped daemon -->
176      <action>
177          <name>telescoped_status</name>
178          <shell>ps x | grep -c "telescoped$$"</shell>
179          <timeout>$TIME_SHORT</timeout>
180          <choose>
181              <when>
182                  <eval><![CDATA[[
183                      $STDOUT1==0
184                  ]]></eval>
185          <choose>
```

```

186      <when>
187          <eval><! [CDATA [
188              ! is_file($FS_PID_TELESCOPED) &&
189              ! is_file($FS_PID_RUND_TELESCOPED)
190          ]]></eval>
191          <send>update HK_TJO_TELESCOPED_STATUS=STOP</send>
192      </when>
193      <otherwise>
194          <send>update HK_TJO_TELESCOPED_STATUS=ERROR</send>
195      </otherwise>
196  </choose>
197 </when>
198 <when>
199     <eval><! [CDATA [
200         $STDOUT1==1
201     ]]></eval>
202     <send>update HK_TJO_TELESCOPED_STATUS=HALF_RUN</send>
203 </when>
204 <when>
205     <eval><! [CDATA [
206         $STDOUT1==2
207     ]]></eval>
208     <shell>ps x | grep -c "rund telescoped$$"</shell>
209     <choose>
210         <when>
211             <eval><! [CDATA [
212                 $STDOUT1==1
213             ]]></eval>
214             <send>update HK_TJO_TELESCOPED_STATUS=RUN</send>
215         </when>
216         <otherwise>
217             <send>update HK_TJO_TELESCOPED_STATUS=ERROR</send>
218         </otherwise>
219     </choose>
220 </when>
221 <otherwise>
222     <send>update HK_TJO_TELESCOPED_STATUS=ERROR</send>
223 </otherwise>
224 </choose>
225 </action>
226 <!-- Check csimcd daemon -->
227 <action>
228     <name>csimcd_status</name>
229     <shell>ps x | grep -c "csimcd.*$PORT_CSIMCD$$"</shell>
230     <timeout>$TIME_SHORT</timeout>
231     <choose>
232         <when>
233             <eval><! [CDATA [
234                 $STDOUT1==0
235             ]]></eval>
236             <choose>
237                 <when>
238                     <eval><! [CDATA [
239                         ! is_file($FS_PID_CSIMCD) &&
240                         ! is_file($FS_PID_RUND_CSIMCD)
241                     ]]></eval>
242                     <send>update HK_TJO_CSIMCD_STATUS=STOP</send>
243                 </when>
244                 <otherwise>
245                     <send>update HK_TJO_CSIMCD_STATUS=ERROR</send>
246                 </otherwise>
247             </choose>
248         </when>
249     <when>
250         <eval><! [CDATA [
251             $STDOUT1==1
252         ]]></eval>
253         <send>update HK_TJO_CSIMCD_STATUS=HALF_RUN</send>
254     </when>
255     <when>
256         <eval><! [CDATA [
257             $STDOUT1==2
258         ]]></eval>
259         <shell>ps x | grep -c "rund csimcd.*$PORT_CSIMCD$$"</shell>
260         <choose>
261             <when>
262                 <eval><! [CDATA [
263                     $STDOUT1==1
264                 ]]></eval>
265                 <send>update HK_TJO_CSIMCD_STATUS=RUN</send>
266             </when>
267             <otherwise>
268                 <send>update HK_TJO_CSIMCD_STATUS=ERROR</send>

```

```

269          </otherwise>
270          </choose>
271      </when>
272      <otherwise>
273          <send>update HK_TJO_CSIMCD_STATUS=ERROR</send>
274      </otherwise>
275      </choose>
276  </action>
277 </actions>
278 </root>

```

12.10.6. dome.xml

```

1  <?xml version="1.0" encoding="UTF-8" ?>
2  <!--
3
4  /---\ - --\ --- - --\ | ---\ / ---\ / ---\ / ---\ / ---\ / ---\ / ---\
5  | | | | , \ / | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
6  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
7  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
8  \---/| .--/ \---\ | | | | | | | | | | | | | | | | | | | | | | | | | | | |
9
10 OpenROCS: Open Robotic Observatory Control System
11 Copyright (C) 2011-2014 by Institut d'Estudis Espacials de Catalunya (IEEC)
12 More information in http://www.ieec.cat or iec@ieec.cat
13
14 This program is free software: you can redistribute it and/or modify
15 it under the terms of the GNU General Public License as published by
16 the Free Software Foundation, either version 3 of the License, or
17 (at your option) any later version.
18
19 This program is distributed in the hope that it will be useful,
20 but WITHOUT ANY WARRANTY; without even the implied warranty of
21 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
22 GNU General Public License for more details.
23
24 You should have received a copy of the GNU General Public License
25 along with this program. If not, see <http://www.gnu.org/licenses/>.
26 -->
27 <root>
28   <actions>
29     <!-- Check dome status -->
30     <action>
31       <name>status</name>
32       <php>$SYS_DOME_STATUS </php>
33       <choose>
34         <when>
35           <eval><! [CDATA [
36             $HK_DOME_CIRCUITS_POWER_STATUS != $HK_DOME_CIRCUITS_POWER_GOAL
37           ]]></eval>
38           <send>update SYS_DOME_STATUS=ERROR</send>
39         </when>
40         <when>
41           <eval><! [CDATA [
42             $HK_DOME_LIGHT_POWER_STATUS != $HK_DOME_LIGHT_POWER_GOAL
43           ]]></eval>
44           <send>update SYS_DOME_STATUS=WARNING</send>
45         </when>
46         <when>
47           <eval><! [CDATA [
48             $HK_DOME_CIRCUITS_POWER_STATUS == $HK_DOME_CIRCUITS_POWER_GOAL &&
49             $HK_DOME_LIGHT_POWER_STATUS == $HK_DOME_LIGHT_POWER_GOAL
50           ]]></eval>
51           <send>update SYS_DOME_STATUS=GOOD</send>
52         </when>
53         <otherwise>
54           <send>update SYS_DOME_STATUS=UNKNOWN</send>
55         </otherwise>
56       </choose>
57       <!-- Log when STATUS changes -->
58       <choose>
59         <when>
60           <eval><! [CDATA [
61             $SYS_DOME_STATUS != $STDOUT
62           ]]></eval>
63           <!-- Define the logging level -->
64           <choose>
65             <when>

```

```

66      <eval><![CDATA[
67          $SYS_DOME_STATUS== 'ERROR ,
68      ]]></eval>
69      <log>$LOG_N2 Setting SYS_DOME_STATUS=$SYS_DOME_STATUS</log>
70  </when>
71  <when>
72      <eval><![CDATA[
73          $SYS_DOME_STATUS== 'WARNING ,
74      ]]></eval>
75      <log>$LOG_N1 Setting SYS_DOME_STATUS=$SYS_DOME_STATUS</log>
76  </when>
77  <when>
78      <eval><![CDATA[
79          $SYS_DOME_STATUS== 'GOOD ,
80      ]]></eval>
81      <log>$LOG_NO Setting SYS_DOME_STATUS=$SYS_DOME_STATUS</log>
82  </when>
83      <otherwise>
84          <log>$LOG_N1 Setting SYS_DOME_STATUS=$SYS_DOME_STATUS</log>
85      </otherwise>
86      </choose>
87  </when>
88  </choose>
89  </action>
90  <!-- Check dome circuits power -->
91  <action>
92      <name>circuits_pdu_status</name>
93      <shell>snmpget -v1 -c public $IP_PDU_NO $OID_PDU_N6</shell>
94  <choose>
95      <when>
96          <eval>$STDOUT4===$PDU_ON</eval>
97          <send>update HK_DOME_CIRCUITS_POWER_STATUS=ON</send>
98      </when>
99      <when>
100         <eval>$STDOUT4===$PDU_OFF</eval>
101         <send>update HK_DOME_CIRCUITS_POWER_STATUS=OFF</send>
102     </when>
103     <otherwise>
104         <send>update HK_DOME_CIRCUITS_POWER_STATUS=ERROR</send>
105     </otherwise>
106     </choose>
107  </action>
108  <!-- Check dome light power -->
109  <action>
110     <name>light_pdu_status</name>
111     <shell>snmpget -v1 -c public $IP_PDU_NO $OID_PDU_N7</shell>
112  <choose>
113      <when>
114          <eval>$STDOUT4===$PDU_ON</eval>
115          <send>update HK_DOME_LIGHT_POWER_STATUS=ON</send>
116      </when>
117      <when>
118          <eval>$STDOUT4===$PDU_OFF</eval>
119          <send>update HK_DOME_LIGHT_POWER_STATUS=OFF</send>
120      </when>
121      <otherwise>
122          <send>update HK_DOME_LIGHT_POWER_STATUS=ERROR</send>
123      </otherwise>
124  </choose>
125  </action>
126  <!-- Check dome flats power -->
127  <action>
128      <name>flats_pdu_status</name>
129      <shell>snmpget -v1 -c public $IP_PDU_NO $OID_PDU_N5</shell>
130  <choose>
131      <when>
132          <eval>$STDOUT4===$PDU_ON</eval>
133          <send>update HK_DOME_FLATS_POWER_STATUS=ON</send>
134      </when>
135      <when>
136          <eval>$STDOUT4===$PDU_OFF</eval>
137          <send>update HK_DOME_FLATS_POWER_STATUS=OFF</send>
138      </when>
139      <otherwise>
140          <send>update HK_DOME_FLATS_POWER_STATUS=ERROR</send>
141      </otherwise>
142      </choose>
143  </action>
144  </actions>
145 </root>
```

12.10.7. meia.xml

```

1  <?xml version="1.0" encoding="UTF-8" ?>
2  <!--
3
4  /_ \ - -- / --- - -- | - \ / / ---/ ---| -- - - \ / _ \
5  | | | | _ \ / - \ | | | | | | | | | | | | | | | | | | | | | |
6  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
7  \---/ | .--/ \---| | | | | | | | | | | | | | | | | | | | | | |
8  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
9
10 OpenROCS: Open Robotic Observatory Control System
11 Copyright (C) 2011-2014 by Institut d'Estudis Espacials de Catalunya (IEEC)
12 More information in http://www.ieec.cat or ieec@ieec.cat
13
14 This program is free software: you can redistribute it and/or modify
15 it under the terms of the GNU General Public License as published by
16 the Free Software Foundation, either version 3 of the License, or
17 (at your option) any later version.
18
19 This program is distributed in the hope that it will be useful,
20 but WITHOUT ANY WARRANTY; without even the implied warranty of
21 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
22 GNU General Public License for more details.
23
24 You should have received a copy of the GNU General Public License
25 along with this program. If not, see <http://www.gnu.org/licenses/>.
26 -->
27 <root>
28     <!-- Monitor: MEIA. Update variables for MEIA -->
29     <actions>
30         <!-- Check the MEIA status -->
31         <action>
32             <name>status</name>
33             <php>$SYS_MEIA_STATUS</php>
34             <choose>
35                 <when>
36                     <eval><! [CDATA [
37                         ($HK_TELRUN_DAEMON_STATUS == 'STOP' &&
38                         (substr($HK_MEIA_CAMERAD_STATUS, -strlen($HK_MEIA_CAMERAD_GOAL)) != <-
39                             $HK_MEIA_CAMERAD_GOAL || 
40                             substr($HK_MEIA_TEMP_STATUS, 0, strlen($HK_MEIA_TEMP_GOAL)) != <-
41                             $HK_MEIA_TEMP_GOAL) || 
42                             ($HK_TELRUN_DAEMON_STATUS != 'STOP' &&
43                             $HK_MEIA_TEMP_STATUS != $HK_MEIA_TEMP_GOAL)
44                         )]]></eval>
45                     <send>update SYS_MEIA_STATUS=ERROR</send>
46                 </when>
47                 <when>
48                     <eval><! [CDATA [
49                         ($HK_TELRUN_DAEMON_STATUS != 'STOP' &&
50                         $HK_MEIA_CAMERAD_STATUS != $HK_MEIA_CAMERAD_GOAL) ||
51                         ($HK_TELRUN_DAEMON_STATUS == 'STOP' &&
52                             $HK_MEIA_TEMP_STATUS != $HK_MEIA_TEMP_GOAL)
53                     )]]></eval>
54                     <send>update SYS_MEIA_STATUS=WARNING</send>
55                 </when>
56                 <when>
57                     <eval><! [CDATA [
58                         $HK_MEIA_CAMERAD_STATUS == $HK_MEIA_CAMERAD_GOAL &&
59                         $HK_MEIA_TEMP_STATUS == $HK_MEIA_TEMP_GOAL
60                     )]]></eval>
61                     <send>update SYS_MEIA_STATUS=GOOD</send>
62                 </when>
63                 <otherwise>
64                     <send>update SYS_MEIA_STATUS=UNKNOWN</send>
65                 </otherwise>
66             </choose>
67             <!-- Log when STATUS changes -->
68             <choose>
69                 <when>
70                     <eval><! [CDATA [
71                         $SYS_MEIA_STATUS != $STDOUT
72                     )]]></eval>
73                     <!-- Define the logging level -->
74                     <choose>
75                         <when>
76                             <eval><! [CDATA [
77                                 $SYS_MEIA_STATUS == 'ERROR',
78                             )]]></eval>
79                             <log>$LOG_N2 Setting SYS_MEIA_STATUS=$SYS_MEIA_STATUS</log>
80                         </when>
81                     </choose>
82                 </when>
83             </choose>
84         </when>
85     </choose>
86 
```

```

79          <when>
80              <eval><![CDATA[
81                  $SYS_MEIA_STATUS=='WARNING',
82              ]]></eval>
83                  <log>$LOG_N1 Setting SYS_MEIA_STATUS=$SYS_MEIA_STATUS</log>
84          </when>
85          <when>
86              <eval><![CDATA[
87                  $SYS_MEIA_STATUS=='GOOD',
88              ]]></eval>
89                  <log>$LOG_NO Setting SYS_MEIA_STATUS=$SYS_MEIA_STATUS</log>
90          </when>
91          <otherwise>
92              <log>$LOG_N1 Setting SYS_MEIA_STATUS=$SYS_MEIA_STATUS</log>
93          </otherwise>
94      </choose>
95  </when>
96  </choose>
97 </action>
98 <!-- Check camerad daemon -->
99 <action>
100    <name>camerad_status</name>
101    <shell>ps x | grep -c "camerad$$"</shell>
102    <timeout>$TIME_SHORT</timeout>
103    <choose>
104        <when>
105            <eval><![CDATA[
106                $STDOUT1==0
107            ]]></eval>
108            <choose>
109                <when>
110                    <eval><![CDATA[
111                        !is_file($FS_PID_CAMERAD) &&
112                        !is_file($FS_PID_RUND_CAMERAD)
113                    ]]></eval>
114                    <send>update HK_MEIA_CAMERAD_STATUS=STOP</send>
115                </when>
116                <otherwise>
117                    <send>update HK_MEIA_CAMERAD_STATUS=ERROR</send>
118                </otherwise>
119            </choose>
120        </when>
121        <when>
122            <eval><![CDATA[
123                $STDOUT1==1
124            ]]></eval>
125            <send>update HK_MEIA_CAMERAD_STATUS=HALF_RUN</send>
126        </when>
127        <when>
128            <eval><![CDATA[
129                $STDOUT1==2
130            ]]></eval>
131            <shell>ps x | grep -c "rund camerad$$"</shell>
132            <choose>
133                <when>
134                    <eval><![CDATA[
135                        $STDOUT1==1
136                    ]]></eval>
137                    <send>update HK_MEIA_CAMERAD_STATUS=RUN</send>
138                </when>
139                <otherwise>
140                    <send>update HK_MEIA_CAMERAD_STATUS=ERROR</send>
141                </otherwise>
142            </choose>
143        </when>
144        <otherwise>
145            <send>update HK_MEIA_CAMERAD_STATUS=ERROR</send>
146        </otherwise>
147    </choose>
148 </action>
149 <!-- Check MEIA CCD temperature -->
150 <action>
151    <name>temp_status</name>
152    <shell>talon_fifo $FS_FIFO_CAMERA_IN "Temperature" $FS_FIFO_CAMERA_OUT ←
153                                $TIME_SHORT "Current temperature:"</shell>
154    <choose>
155        <when>
156            <eval><![CDATA[
157                is_numeric($STDOUT1)
158            ]]></eval>
159            <choose>
160                <when>

```

```

161                               $STDOUT6=='Ramping',
162                         ]]></eval>
163                         <send>update SDP_MEIA_TEMP_VALUE=$STDOUT4</send>
164                         <send>update HK_MEIA_TEMP_STATUS=COOLING</send>
165                   </when>
166                   <when>
167                     <eval><! [CDATA [
168                         $STDOUT6=='AtTarg',
169                   ]]></eval>
170                     <send>update SDP_MEIA_TEMP_VALUE=$STDOUT4</send>
171                     <send>update HK_MEIA_TEMP_STATUS=COOL</send>
172                   </when>
173                   <otherwise>
174                     <send>update SDP_MEIA_TEMP_VALUE=0</send>
175                     <send>update HK_MEIA_TEMP_STATUS=ERROR</send>
176                   </otherwise>
177                 </choose>
178               </when>
179               <when>
180                 <eval><! [CDATA [
181                     strpos($STDOUT,"Fifo to read not found")!==FALSE
182                   ]]></eval>
183                     <send>update SDP_MEIA_TEMP_VALUE=0</send>
184                     <send>update HK_MEIA_TEMP_STATUS=OFF</send>
185               </when>
186               <otherwise>
187                 <send>update SDP_MEIA_TEMP_VALUE=0</send>
188                 <send>update HK_MEIA_TEMP_STATUS=ERROR</send>
189               </otherwise>
190             </choose>
191           </action>
192         <actions>
193       </root>

```

12.10.8. meteo.xml

```

1  <?xml version="1.0" encoding="UTF-8" ?>
2  <!--
3
4  /_ \ - -- / _ \ , _ \ / _ \ / _ \ / _ \ / _ \ / _ \ / _ \ / _ \ / _ \
5  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
6  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
7  \_/_ / \_/_ / \_/_ / \_/_ / \_/_ / \_/_ / \_/_ / \_/_ / \_/_ / \_/_ / \_/_ /
8  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
9
10 OpenROCS: Open Robotic Observatory Control System
11 Copyright (C) 2011-2014 by Institut d'Estudis Espacials de Catalunya (IEEC)
12 More information in http://www.ieec.cat or ieec@ieec.cat
13
14 This program is free software: you can redistribute it and/or modify
15 it under the terms of the GNU General Public License as published by
16 the Free Software Foundation, either version 3 of the License, or
17 (at your option) any later version.
18
19 This program is distributed in the hope that it will be useful,
20 but WITHOUT ANY WARRANTY; without even the implied warranty of
21 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
22 GNU General Public License for more details.
23
24 You should have received a copy of the GNU General Public License
25 along with this program. If not, see <http://www.gnu.org/licenses/>.
-->
26
27 <root>
28   <!-- Monitor: METEO. Update variables related with time and -->
29   <!-- meteorological information -->
30   <actions>
31     <!-- Check the meteorological status (currently, only daemons) -->
32     <action>
33       <name>status</name>
34       <php>$SYS_METEO_STATUS</php>
35       <choose>
36         <when>
37           <eval><! [CDATA [
38             $HK_TELRUN_DAEMON_STATUS=='STOP', &&
39             (substr($HK_METEO_WXD_READ_STATUS,-strlen($HK_METEO_WXD_READ_GOAL))!=
39              $HK_METEO_WXD_READ_GOAL ||
40              substr($HK_METEO_WXD_WRITE_STATUS,-strlen($HK_METEO_WXD_WRITE_GOAL))!=
40              $HK_METEO_WXD_WRITE_GOAL ||

```

```

41                      substr($HK_METEO_GPSD_STATUS,-strlen($HK_METEO_GPSD_GOAL))!=<→
42                      $HK_METEO_GPSD_GOAL)
43                  ]]></eval>
44                  <send>update SYS_METEO_STATUS=ERROR</send>
45              </when>
46              <when>
47                  <eval><! [CDATA [
48                      $HK_TELRUN_DAEMON_STATUS=='RUN' &&
49                      ($HK_METEO_GPSD_GOAL!=$HK_METEO_GPSD_STATUS ||
50                      $HK_METEO_WXD_READ_GOAL!=$HK_METEO_WXD_READ_STATUS ||
51                      $HK_METEO_WXD_WRITE_GOAL!=$HK_METEO_WXD_WRITE_STATUS)
52                  ]]></eval>
53                  <send>update SYS_METEO_STATUS=WARNING</send>
54              </when>
55              <when>
56                  <eval><! [CDATA [
57                      $HK_METEO_GPSD_GOAL==$HK_METEO_GPSD_STATUS &&
58                      $HK_METEO_WXD_READ_GOAL==$HK_METEO_WXD_READ_STATUS &&
59                      $HK_METEO_WXD_WRITE_GOAL==$HK_METEO_WXD_WRITE_STATUS
60                  ]]></eval>
61                  <send>update SYS_METEO_STATUS=GOOD</send>
62              </when>
63              <otherwise>
64                  <send>update SYS_METEO_STATUS=UNKNOWN</send>
65              </otherwise>
66          </choose>
67          <!-- Log when STATUS changes -->
68          <choose>
69              <when>
70                  <eval><! [CDATA [
71                      $SYS_METEO_STATUS!=$STDOUT
72                  ]]></eval>
73                  <!-- Define the logging level -->
74                  <choose>
75                      <when>
76                          <eval><! [CDATA [
77                              $SYS_METEO_STATUS=='ERROR',
78                          ]]></eval>
79                          <log>$LOG_N2 Setting SYS_METEO_STATUS=$SYS_METEO_STATUS</log>
80                      </when>
81                      <when>
82                          <eval><! [CDATA [
83                              $SYS_METEO_STATUS=='WARNING',
84                          ]]></eval>
85                          <log>$LOG_N1 Setting SYS_METEO_STATUS=$SYS_METEO_STATUS</log>
86                      </when>
87                      <when>
88                          <eval><! [CDATA [
89                              $SYS_METEO_STATUS=='GOOD',
90                          ]]></eval>
91                          <log>$LOG_NO Setting SYS_METEO_STATUS=$SYS_METEO_STATUS</log>
92                      </when>
93                      <otherwise>
94                          <log>$LOG_N1 Setting SYS_METEO_STATUS=$SYS_METEO_STATUS</log>
95                      </otherwise>
96                  </choose>
97              </when>
98          </choose>
99      </action>
100      <!-- Check wxd.php daemon -->
101      <action>
102          <name>wxd_write_status</name>
103          <shell>ps x | grep -c "wxd.php .* -e 0$$"</shell>
104          <timeout>$TIME_SHORT</timeout>
105          <choose>
106              <when>
107                  <eval><! [CDATA [
108                      $STDOUT1==0
109                  ]]></eval>
110                  <choose>
111                      <when>
112                          <eval><! [CDATA [
113                              ! is_file($FS_PID_RUND_WXD_WRITE)
114                          ]]></eval>
115                          <send>update HK_METEO_WXD_WRITE_STATUS=STOP</send>
116                      </when>
117                      <otherwise>
118                          <send>update HK_METEO_WXD_WRITE_STATUS=ERROR</send>
119                      </otherwise>
120                  </choose>
121              </when>
122              <when>
123                  <eval><! [CDATA [

```

```

123          $STDOUT1==1
124      ]]></eval>
125      <send>update HK_METEO_WXD_WRITE_STATUS=HALF_RUN</send>
126  </when>
127  <when>
128      <eval><! [CDATA [
129          $STDOUT1==2
130      ]]></eval>
131      <shell>ps x | grep -c "rund wxd.php .* -e 0$$"</shell>
132      <choose>
133          <when>
134              <eval><! [CDATA [
135                  $STDOUT1==1
136              ]]></eval>
137              <send>update HK_METEO_WXD_WRITE_STATUS=RUN</send>
138          </when>
139          <otherwise>
140              <send>update HK_METEO_WXD_WRITE_STATUS=ERROR</send>
141          </otherwise>
142      </choose>
143  </when>
144  <otherwise>
145      <send>update HK_METEO_WXD_WRITE_STATUS=ERROR</send>
146  </otherwise>
147      </choose>
148  </action>
149  <!-- Check wxd daemon -->
150  <action>
151      <name>wxd_read_status</name>
152      <shell>ps x | grep -c "wxd .* -$$"</shell>
153      <timeout>$TIME_SHORT</timeout>
154      <choose>
155          <when>
156              <eval><! [CDATA [
157                  $STDOUT1==0
158              ]]></eval>
159              <choose>
160                  <when>
161                      <eval><! [CDATA [
162                          ! is_file($FS_PID_WXD_READ) &&
163                          ! is_file($FS_PID_RUND_WXD_READ)
164                      ]]></eval>
165                      <send>update HK_METEO_WXD_READ_STATUS=STOP</send>
166                  </when>
167                  <otherwise>
168                      <send>update HK_METEO_WXD_READ_STATUS=ERROR</send>
169                  </otherwise>
170              </choose>
171          </when>
172          <when>
173              <eval><! [CDATA [
174                  $STDOUT1==1
175              ]]></eval>
176              <send>update HK_METEO_WXD_READ_STATUS=HALF_RUN</send>
177          </when>
178          <when>
179              <eval><! [CDATA [
180                  $STDOUT1==2
181              ]]></eval>
182              <shell>ps x | grep -c "rund wxd .* -$$"</shell>
183              <choose>
184                  <when>
185                      <eval><! [CDATA [
186                          $STDOUT1==1
187                      ]]></eval>
188                      <send>update HK_METEO_WXD_READ_STATUS=RUN</send>
189                  </when>
190                  <otherwise>
191                      <send>update HK_METEO_WXD_READ_STATUS=ERROR</send>
192                  </otherwise>
193              </choose>
194          </when>
195          <otherwise>
196              <send>update HK_METEO_WXD_READ_STATUS=ERROR</send>
197          </otherwise>
198      </choose>
199  </action>
200  <!-- Check gpsd daemon -->
201  <action>
202      <name>gpsd_status</name>
203      <shell>ps ax | grep -c "gpsd -fsac$$"</shell>
204      <timeout>$TIME_SHORT</timeout>
205      <choose>
```

```
206
207     <when>
208         <eval><! [CDATA [
209             $STDOUT1==0
210         ]]></eval>
211         <choose>
212             <when>
213                 <eval><! [CDATA [
214                     !is_file($FS_PID_GPSD) &&
215                     !is_file($FS_PID_RUND_GPSD)
216                 ]]></eval>
217                 <send>update HK_METEO_GPSD_STATUS=STOP</send>
218             </when>
219             <otherwise>
220                 <send>update HK_METEO_GPSD_STATUS=ERROR</send>
221             </otherwise>
222         </choose>
223     </when>
224     <when>
225         <eval><! [CDATA [
226             $STDOUT1==1
227         ]]></eval>
228         <send>update HK_METEO_GPSD_STATUS=HALF_RUN</send>
229     </when>
230     <when>
231         <eval><! [CDATA [
232             $STDOUT1==2
233         ]]></eval>
234         <shell>ps ax | grep -c "rund gpsd -fsac$$"</shell>
235         <choose>
236             <when>
237                 <eval><! [CDATA [
238                     $STDOUT1==1
239                 ]]></eval>
240                 <send>update HK_METEO_GPSD_STATUS=RUN</send>
241             </when>
242             <otherwise>
243                 <send>update HK_METEO_GPSD_STATUS=ERROR</send>
244             </otherwise>
245         </choose>
246     </when>
247     <otherwise>
248         <send>update HK_METEO_GPSD_STATUS=ERROR</send>
249     </otherwise>
250   </actions>
251 </root>
```

12.10.9. telrun.xml

```
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <!--
3
4   /--\ - -- \----|----\ /----/----|----\ /----\ /----\ /----\ /----\
5   | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
6   | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
7   \---/ .--\ \---/ | | | | | | | | | | | | | | | | | | | | | | | | | | | |
8   | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
9
10 OpenROCS: Open Robotic Observatory Control System
11 Copyright (C) 2011-2014 by Institut d'Estudis Espacials de Catalunya (IEEC)
12 More information in http://www.ieec.cat or ieec@ieec.cat
13
14 This program is free software: you can redistribute it and/or modify
15 it under the terms of the GNU General Public License as published by
16 the Free Software Foundation, either version 3 of the License, or
17 (at your option) any later version.
18
19 This program is distributed in the hope that it will be useful,
20 but WITHOUT ANY WARRANTY; without even the implied warranty of
21 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
22 GNU General Public License for more details.
23
24 You should have received a copy of the GNU General Public License
25 along with this program. If not, see <http://www.gnu.org/licenses/>.
26 -->
27 <root>
28   <!-- Actions: TELRUN. Update variables for telrun -->
```

```

29      <actions>
30          <!-- Check the TELRUN status -->
31          <action>
32              <name>status</name>
33              <php>$SYS_TELRUN_STATUS </php>
34              <choose>
35                  <when>
36                      <eval><! [CDATA [
37                          substr($HK_TELRUN_DAEMON_STATUS,-strlen($HK_TELRUN_DAEMON_GOAL)) != -->
38                                         $HK_TELRUN_DAEMON_GOAL ||
39                                         ($HK_TELRUN_NLINES_STATUS == 'ERROR' &&
40                                         $HK_TELRUN_NOBS_STATUS == 'ERROR' &&
41                                         $HK_TELRUN_NCAL_STATUS == 'ERROR')
42                      ]]></eval>
43                      <send>update SYS_TELRUN_STATUS=ERROR </send>
44                  </when>
45                  <when>
46                      <eval><! [CDATA [
47                          $HK_TELRUN_DAEMON_GOAL != $HK_TELRUN_DAEMON_STATUS ||
48                          $HK_TELRUN_NLINES_STATUS == 'ERROR' ||
49                          $HK_TELRUN_NOBS_STATUS == 'ERROR' ||
50                          $HK_TELRUN_NCAL_STATUS == 'ERROR'
51                      ]]></eval>
52                      <send>update SYS_TELRUN_STATUS=WARNING </send>
53                  </when>
54                  <when>
55                      <eval><! [CDATA [
56                          $HK_TELRUN_DAEMON_GOAL == $HK_TELRUN_DAEMON_STATUS &&
57                          $HK_TELRUN_NLINES_STATUS != 'ERROR' &&
58                          $HK_TELRUN_NOBS_STATUS != 'ERROR' &&
59                          $HK_TELRUN_NCAL_STATUS != 'ERROR'
60                      ]]></eval>
61                      <send>update SYS_TELRUN_STATUS=GOOD </send>
62                  </when>
63                  <otherwise>
64                      <send>update SYS_TELRUN_STATUS=UNKNOWN </send>
65                  </otherwise>
66              </choose>
67              <!-- Log when STATUS changes -->
68              <choose>
69                  <when>
70                      <eval><! [CDATA [
71                          $SYS_TELRUN_STATUS != $STDOUT
72                      ]]></eval>
73                      <!-- Define the logging level -->
74                      <choose>
75                          <when>
76                              <eval><! [CDATA [
77                                  $SYS_TELRUN_STATUS == 'ERROR',
78                              ]]></eval>
79                              <log>$LOG_N2 Setting SYS_TELRUN_STATUS=$SYS_TELRUN_STATUS </log>
80                          <when>
81                              <eval><! [CDATA [
82                                  $SYS_TELRUN_STATUS == 'WARNING',
83                              ]]></eval>
84                              <log>$LOG_N1 Setting SYS_TELRUN_STATUS=$SYS_TELRUN_STATUS </log>
85                          <when>
86                              <eval><! [CDATA [
87                                  $SYS_TELRUN_STATUS == 'GOOD',
88                              ]]></eval>
89                              <log>$LOG_NO Setting SYS_TELRUN_STATUS=$SYS_TELRUN_STATUS </log>
90                          <when>
91                              <otherwise>
92                                  <log>$LOG_N1 Setting SYS_TELRUN_STATUS=$SYS_TELRUN_STATUS </log>
93                              </otherwise>
94                          </choose>
95                      </when>
96                  </choose>
97              </action>
98              <!-- Check telrun daemon -->
99              <action>
100                  <name>daemon_status</name>
101                  <shell>ps x | grep -c "telrun$$" </shell>
102                  <timeout>$TIME_SHORT </timeout>
103                  <choose>
104                      <when>
105                          <eval><! [CDATA [
106                              $STDOUT1 == 0
107                          ]]></eval>
108                          <choose>
109                              <when>
```

```

111      <eval><! [CDATA [
112          ! is_file($FS_PID_TELRUN) &&
113          ! is_file($FS_PID_RUND_TELRUN)
114      ]]></eval>
115          <send>update HK_TELRUN_DAEMON_STATUS=STOP</send>
116      </when>
117      <otherwise>
118          <send>update HK_TELRUN_DAEMON_STATUS=ERROR</send>
119      </otherwise>
120      </choose>
121  </when>
122  <when>
123      <eval><! [CDATA [
124          $STDOUT1==1
125      ]]></eval>
126          <send>update HK_TELRUN_DAEMON_STATUS=HALF_RUN</send>
127  </when>
128  <when>
129      <eval><! [CDATA [
130          $STDOUT1==2
131      ]]></eval>
132          <shell>ps x | grep -c "rund telescoped$$"</shell>
133      <choose>
134          <when>
135              <eval><! [CDATA [
136                  $STDOUT1==1
137              ]]></eval>
138              <send>update HK_TELRUN_DAEMON_STATUS=RUN</send>
139          </when>
140          <otherwise>
141              <send>update HK_TELRUN_DAEMON_STATUS=ERROR</send>
142          </otherwise>
143      </choose>
144  </when>
145  <otherwise>
146      <send>update HK_TELRUN_DAEMON_STATUS=ERROR</send>
147  </otherwise>
148      </choose>
149  </action>
150  </actions>
151 </root>

```


13. Bibliografía

1. Página principal del Proyecto Fedora
<http://fedoraproject.org/>
2. Geany - A text editor using the GTK2 toolkit
<http://www.geany.org/>
3. txt2tags - ONE source, MULTI targets
<http://txt2tags.org/>
4. TeX Live - TeX Users Group
<http://tug.org/texlive/>
5. GIMP - The GNU Image Manipulation Program
<http://www.gimp.org/>
6. IEEC - Institut d'Estudis Espacials de Catalunya
<http://www.ieec.cat/es/investigacion/oadm/>
7. TJO - Telescopio Joan Oró
<http://www.oadm.cat/es/content/45/tjo.htm>
8. OAdM - Observatorio Astronómico del Montsec
<http://www.oadm.cat/es/content/1/oadm.htm>
9. The artificial night sky brightness mapped from DMSP Operational Linescan System measurements
P. Cinzano (1), F. Falchi (1), C.D. Elvidge (2), K.E. Baugh (2)
((1) Dipartimento di Astronomia Padova, Italy, (2) Solar-Terrestrial Physics Division, NOAA National Geophysical Data Center, Boulder, CO)
Monthly Notices of the Royal Astronomical Society, 318, 641-657 (2000)
<http://www.lightpollution.it/cinzano/papers.html>
http://www.lightpollution.it/cinzano/download/mnras_paper.pdf
10. Talon - Observatory Control Software
<http://sourceforge.net/projects/observatory/>
11. XEphem - Astronomical Software Ephemeris
<http://www.clearskyinstitute.com/xephem/>
12. Linux Journal - Linux, Talon and Astronomy
<http://www.linuxjournal.com/article/6673>
13. ZeroC Object-Oriented middleware
<http://www.zeroc.com/index.html>
14. Robot software - Wikipedia, the free encyclopedia
http://en.wikipedia.org/wiki/Robot_software
15. Robotic telescope - Wikipedia, the free encyclopedia
http://en.wikipedia.org/wiki/Robotic_telescope
16. RTS2: open source standard and package for autonomous observatory
<http://www.rts2.org/>

17. RTS2: howto:intergation_of_a_davis_meteo_station (RTS2 Wiki)
http://rts2.org/wiki/doku.php?id=howto:intergation_of_a_davis_meteo_station
18. ASCOM (standard) - Wikipedia, the free encyclopedia
http://en.wikipedia.org/wiki/ASCOM_%28standard%29
19. ASCOM - Standards for Astronomy
<http://ascom-standards.org/>
20. INDI Library provides a framework for control and automation of astronomical instruments.
<http://www.indilib.org/>
21. The Orocос Project - Smarter control in robotics & automation!
<http://www.orocos.org/>
22. ROS.org - Powering the world's robots
<http://www.ros.org/>
23. The SaltOS Project:
<http://www.saltos.net/>
24. PHP Project
<http://www.php.net/>
25. PHP Documentation
<http://www.php.net/manual/en/>
26. OpenROCS - Open Robotic Observatory Control System
<http://sourceforge.net/projects/openrocs/>
27. Campbell Scientific
<http://www.campbellsci.com/>
28. PBCdIComm
<https://engineering.arm.gov/~choudhury/PbCdlComm/index.htm>
29. Davis Vantage Pro2
<http://www.davisnet.com/weather/products/vantage-pro-professional-weather-stations.asp>
30. Vaisala
<http://www.vaisala.com/en/Pages/default.aspx>
31. Eigenbrodt GmbH & Co. KG
<http://www.eigenbrodt.de/en/products/meteorologie/precipitation-sensor-monitor.html>
32. Cyanogen Imaging Products from Diffraction Limited
http://www.cyanogen.com/cloud_dl.php
33. Ingesco Lightning Solutions
<http://www.ingesco.com/en/products/preventive-protection/preventive-protection-products>
34. APC Switched Rack PDU
http://www.apc.com/products/resource/include/techspec_index.cfm?base_sku=AP7920
35. SuperWASP Project:
<http://www.superwasp.org/>

36. SQT Project - Super Wasp Qatar Follow-up Telescopes
<http://www.ieec.cat/es/project/sqt/>
37. HHVM - HipHop Virtual Machine
<http://hhvm.com/>
38. GitHub facebook/hhvm repository
<https://github.com/facebook/hhvm>
39. ACM International Conference on Object Oriented Programming Systems, Languages, and Applications (OOPSLA)
<https://www.facebook.com/publications/483645601777040/>
40. FasterCGI with HHVM - Performance - Fibonacci
<http://hhvm.com/blog/1817/fastercgi-with-hhvm>
41. HippyVM - An implementation of the PHP language in RPython
<http://hippyvm.com/>
42. GitHub hippyvm/hippyvm repository
<https://github.com/hippyvm/hippyvm>
43. XML From Wikipedia, the free encyclopedia
<http://en.wikipedia.org/wiki/XML>
44. Extensible Markup Language (XML)
<http://www.w3.org/XML/>